FTPEXT Working Group
Internet-Draft
Expiration Date: November 28, 2002

G. Lundberg
WU-FTPD Development Group

May 2002

FTP Data Connection Assurance
draft-ietf-ftpext-data-connection-assurance-00.txt

Status of this Memo

Copyright Notice

Abstract

This document specifies an extension to the File Transfer Protocol (FTP) by which a user and server can exchange data port connection information which may then be used to provide connection assurance.

Two new commands are introduced. Through use of these commands and their replies, the user and server exchange information allowing verification of the socket addresses for a data connection prior to actual data transmission over the connection.

Implementation of this extension is RECOMMENDED.

Table of Contents

1. Introduction

The intention of this protocol extension is to address the security issue involved with the problem of a race condition in the existing File Transfer Protocol [DS, CERT2]. This condition allows an attacker the ability to receive unauthorized transmissions from, or transmit unauthorized information to, the passive party of a file transfer. The attack makes use of the timing window between when the passive socket is prepared and when an active connection to that passive socket actually occurs.

The security industry identifier for this issue is CVE-1999-0351 [CVE].

The problem is fairly well known within the FTP community and has been discussed for some time [DS]. Since it does not pose a direct threat to the security of hosts on the Internet, and generally presents only limited inconvenience to users, little attention has been paid to the issue. It is, however, a growing concern to a number of private, corporate and governmental users who desire assurance that their information is delivered only to the intended recipients yet who are unwilling to sacrifice availability or inter-operability by use of cryptographic methods [RFC2228, MURRAY]. The fact that the FTP makes no attempt to prevent, or even provide a means to detect, these events is unacceptable.

Several independent implementations of attacks making use of this window are known to exist [KS, JG], and at least one event has been claimed to have occurred against a vendor site to obtain unauthorized copies of potentially proprietary information. To date, all known attacks depend upon weaknesses in TCP implementations, but with the relatively small number of TCP ports and advances in distributed attack methods this condition cannot be expected to continue.

To fully understand the problem, and the solution presented, a basic understanding of some features of the FTP is required.

The FTP model [RFC959] uses two communication channels: the data connection is responsible for exchanging files and other data; the control connection provides control and synchronization of the data connection and transmissions over it. While the control connection exists throughout the lifetime of an FTP session, data connections are often transitory, existing only for the duration of a single file transfer.

The FTP model is usually presented as two processes: the User-FTP implementation and the Server-FTP implementation. Each of these processes are further separated into the protocol interpreter (PI)

and the data transfer process (DTP). A Server-FTP always implements both a PI and a DTP. The User-FTP, however, may implement only a PI; using an external process as its DTP (as is the case, for example, in a server-to-server transfer).

Existing FTP commands [RFC959, RFC1639, RFC2428] allow the user to determine which end-point has responsibility for actively initiating the data connection. The other end-point passively awaits this connection. To cause the Server-DTP to operate in passive mode, the user sends a PASV, LPSV or EPSV command, requesting the Server-DTP create a passive socket and report the address assigned. To cause the Server-DTP to operate in active mode, the user creates (or obtains through other means) a passive socket and communicates its address to the Server-FTP process through the PORT, LPRT or EPRT commands.

The user actively initiates the data connection immediately prior to issuing the data transfer command. (It may do so any time after receipt of the reply to the PASV, LPSV or EPSV command, and must do so prior to issuing the data transfer command.) The Server-FTP process must initiate the data connection upon receipt of the data transfer command and prior to issuing a reply to that command.

The delay involved, between the creation of the passive socket and establishment of the data connection using it, presents a window during which an attacker can actively establish a connection to the passive socket.

Unfortunately, the FTP provides no means to assure that the active socket is the one intended; the passive participant simply accepts the first connection presented. While all parties have knowledge of the address assigned to the passive socket, only the process actively initiating the connection knows the address of the active socket. Without some means of communicating this information to all parties, there is insufficient information to judge whether the connection originated from the intended participant.

The protocol presented is intended to provide just that: a means to exchange active socket address information.

In the development of the protocol specified in this document, several alternatives were considered.

The STAT command [RFC959] could be issued by the user to discover the address of the active socket. The format of the reply to the STAT command, however, is not well defined; to be usable, a consistent, machine readable response is needed. Requiring the reply to the STAT command to provide the necessary information in a usable fashion

would cause inter-operation problems with existing implementations, and is only a partial solution since it does not address the needs of the Server-FTP process when operating in passive mode.

Some existing Server-FTP implementations attempt to reduce the impact of the problem by placing requirements upon which remote socket addresses they will accept [GL]. These requirements violate the spirit, if not the letter, of the FTP specifications, negatively impacting its usability, and can raise complex configuration issues for server administrators. In addition, these methods are only a partial solution since they do not address the needs of the user.

[DS] suggests exchanging verification information over the control and data connections, however doing so would prevent inter-operability with existing implementations. Existing specifications [RFC2228, MURRAY] can provide just such a solution but are not widely deployed and, to assure connection integrity, do not inter-operate with existing (non-cryptographic) implementations.

What is required is a solution which addresses the problem for all parties, operates with little added overhead and minimal (optimally, no) configuration requirements, does not break inter-operability with existing implementations, and which is straightforward to deploy, without the processing overhead and potential legal issues involved with cryptographic methods.

The new commands presented here provide for the exchange the active socket address between the user and the Server-FTP process. This presents both parties (or, in the case of server-to-server transfers, all three parties) with the information necessary to evaluate the connection end-points prior to commencing transmission.

When reading the following specifications, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [RFC2119].

2. ACTIVE ESTABLISH (ESTA)

The user should issue the ESTA command before issuing any other data transfer command when the Server-DTP is in active mode and a usable data connection is not already present.

This command instructs the Server-DTP to immediately establish a connection from its data port to the currently selected passive address, and to report back to the user the address information for the active socket actually used on the new connection.

The active socket on the connection is the local address. In [POSIX], for example, this is the value returned via the getsockname function; which may differ from the address requested by the bind function.

The user must have already prepared a passive socket to accept the connection, and communicated the address information for that socket to the Server-FTP process using the PORT, LPRT or EPRT command. For a server-to-server transfer, this means the user must have already successfully instructed the other, passive Server-DTP to prepare the passive socket.

If the Server-DTP has already accepted an established data connection to the passive socket, it should report the local address for that connection. Thus, the ESTA command is repeatable, and may be issued to either server in a server-to-server transfer at any time that the user wishes the server to report the local address it observes on the fully established data connection.

The establishment of the connection may take some time, and the Server-DTP may make several attempts before reporting a failure. The implementation should include a time out limiting the duration of the connection attempts. The Server-FTP process should be capable of accepting the ABOR command during this period, and should interpret the ABOR command as a request to cease the connection attempt or close the connection if the attempt has already succeeded.

The ESTA command is analogous to the PASV, LPSV and EPSV commands, and returns a result formatted the same as that which would be produced by the EPSV command [RFC2428] (even if the EPSV command is not implemented).

The format of the ESTA command is:

```
ESTA <CRLF>
```

The ESTA command takes no arguments.

Possible replies to the ESTA command, and their meanings, include:

```
225 Data connection open; no transfer in progress.
421 Service not available, closing control connection.
425 Can't open data connection.
500 Syntax error, command unrecognized.
501 Syntax error in parameters or arguments.
502 Command not implemented.
520 Requested action not taken; DTP is in passive mode.
```

With the exception of response code 225, the User-FTP process must not depend upon the actual text (if any) included with the reply.

A Server-FTP process which does not implement the ESTA command will reply with either response code 500 or 502. For compatibility with existing implementations, the User-FTP process must be prepared for this reply. It should be possible to proceed with the data transfer. Local site policy, however, may dictate refusal to continue communication with implementations which do not support ESTA.

The response code 520 indicates that the Server-DTP was unable to accept a connection from the passive socket because it is operating in passive mode and does not have the address of a remote passive socket to connect to. This should not occur if there is already a fully established data connection.

The response code 425 indicates that the Server-DTP was unable to establish a connection to the passive socket. This should not occur if there is already a fully established data connection. In [POSIX], for example, this would occur when the connect function returns an error indication; possible causes might include a refused connection, an unreachable passive socket address, time-out, resource depletion, or internal implementation errors.

Response code 225 indicates that the Server-DTP has established a connection to the passive socket, or already had a fully established data connection.

The Server-FTP process must format the text of reply for response code 225 in a manner similar to a positive completion reply to the EPSV command (even if the EPSV command is not actually implemented). Unlike the EPSV reply, the Server-FTP process must include the full protocol address information actually observed on the data connection for the local socket in the ESTA report.

Upon receipt of reply 225, the user's DTP should accept the connection from the passive socket and then the User-FTP process should compare the information provided with the reply against that obtained locally from the established connection.

If the two do not exactly match, the user should terminate the data connection by issuing the ABOR command, and should not proceed to issue a data transfer command.

For a server-to-server transfer, the information provided in the response to ESTA should be compared with the information provided in the response to the ESTP command. For the User-FTP process, the information provided in the response to ESTA should be compared

against the remote address observed on the fully established data connection. In [POSIX], for example, this is the value returned via the address argument to the accept function.

The format of the response code 225 reply is:

225 <SP> <text> <SP> (<d><proto><d><addr><d><port><d>) <CRLF>

The unformatted text portion of the reply (<text>) may be any message, and may be omitted. If present, it must be separated from the formatted portion of the reply by at least one space character.

The formatted portion of the reply must be enclosed in parentheses, formatted exactly as the arguments to the ESTP command (below), and has the same meaning.

Following the separating space and parenthesis, a delimiter character (<d>) must be specified. The delimiter character must be one octet in range 33 to 126, inclusive. The character "|" (ASCII 124) is recommended; unless it coincides with a character needed to encode the arguments.

Note that the discussion which follows presumes the use of TCP over an Internet Protocol but should not be taken as a requirement that only those protocols may be used: from the point of view of the ESTA command, any protocol is acceptable. The number and meaning of any arguments which follow <proto> is protocol-specific; this memo specifies the arguments for TCP over the Internet Protocols in use at the time of its writing.

The protocol argument (<proto>) must be an address family number assigned by the IANA. This number indicates the protocol to be used (and, implicitly, the number, meaning, and maximum lengths, of the remaining arguments).

The network address argument (<addr>) is a protocol-specific string representation of the network address for the active socket observed on the connection. Refer to Annex B for the formats required for the Internet Protocols.

The port number argument (<port>) must be the string representation of the number of the TCP port used by the active socket observed on the connection.

3. PASSIVE ESTABLISH (ESTP)

The user should issue the ESTP command before issuing any other data transfer command when the Server-DTP has been placed in passive mode

and a usable data connection is not already present.

This command instructs the Server-DTP to accept a connection from the passive socket listening on its data port, and to report back to the user the address information for the active socket observed on the new connection.

The active socket observed on the connection is the remote address. In [POSIX], for example, this is the value returned via the address argument to the accept function.

The user must have already established an active connection to the Server-DTP using the address and port information previously obtained from the PASV, LPSV or EPSV command. For a server-to-server transfer, this means the user must have already successfully instructed the other, active Server-DTP to establish the connection.

If the Server-DTP has already accepted an established data connection from the passive socket, it should report the remote address for that connection. Thus, the ESTP command is repeatable, and may be issued to either server in a server-to-server transfer at any time that the user wishes the server to report the remote address it observes on the fully established data connection.

The Server-DTP must not await new connections on the passive socket. The ESTP command indicates the user believes an established connection already exists. If this is not the case, the Server-DTP must immediately report this fact.

The ESTP command is analogous to the PORT, LPRT and EPRT commands, and requires a parameter formatted the same as that which would be provided with the EPRT command [RFC2428] (even if the EPRT command is not implemented). The protocol, network address and port information provided with the command must be that of the active socket.

In a server-to-server transfer, the address of the active socket is returned in the response to the ESTA command. For the User-FTP process, this address is the local information observed on the established active connection. In [POSIX], for example, this is the value returned via the getsockname function; which may differ from the address requested by the bind function.

The format of the ESTP command is:

```
ESTP <SP> <d><proto><d><addr><d><port><d> <CRLF>
```

The ESTP command keyword must be followed by a single space character (ASCII 32). Following the space, a delimiter character (<d>) must be

specified. The delimiter character must be one octet in range 33 to 126, inclusive. The character "|" (ASCII 124) is recommended; unless it coincides with a character needed to encode the arguments.

Note that the command format shown, and the discussion which follows, presume the use of TCP over an Internet Protocol but should not be taken as a requirement that only those protocols may be used: from the point of view of the ESTP command, any protocol is acceptable. The number and meaning of any arguments which follow <proto> is protocol-specific; this memo specifies the arguments for TCP over the Internet Protocols in use at the time of its writing.

The protocol argument (<proto>) must be an address family number assigned by the IANA. This number indicates the protocol to be used (and, implicitly, the number, meaning, and maximum lengths, of the remaining arguments).

The network address argument (<addr>) is a protocol-specific string representation of the network address for the active socket observed on the connection. Refer to Annex B for the formats required for the Internet Protocols.

The port number argument (<port>) must be the string representation of the number of the TCP port used by the active socket observed on the connection.

Possible replies to the ESTP command, and their meanings, include:

225 Data connection open; no transfer in progress.
421 Service not available, closing control connection.
425 Can't open data connection.
500 Syntax error, command unrecognized.
501 Syntax error in parameters or arguments.
502 Command not implemented.
520 Requested action not taken; DTP is in active mode.

With the exception of response code 225, the User-FTP process must not depend upon the actual text (if any) included with the reply.

A Server-FTP process which does not implement the ESTP command will reply with either response code 500 or 502. For compatibility with existing implementations, the User-FTP process must be prepared for this reply. It should be possible to proceed with the data transfer. Local site policy, however, may dictate refusal to continue communication with implementations which do not support ESTP.

The response code 520 indicates that the Server-DTP was unable to accept a connection from the passive socket because it is operating

in active mode and there is no passive socket. This should not occur if there is already a fully established data connection.

The response code 425 indicates that the Server-DTP was unable to accept a connection from the passive socket. This should not occur if there is already a fully established data connection. In [POSIX], for example, this would occur when the accept function returns an error indication; possible causes might include an aborted connection, resource depletion, or internal implementation errors.

Response code 225 indicates that the Server-DTP has accepted a passive data connection, or already had a fully established data connection.

The Server-FTP process must format the text of reply for response code 225 in a manner similar to a positive completion reply to the EPSV command (even if the EPSV command is not actually implemented). Unlike the EPSV reply, the Server-FTP process must include the full protocol address information actually observed on the data connection for the remote socket in the ESTP report.

Upon receipt of reply 225, the User-FTP process should compare the information provided with the reply against that obtained locally from the established connection.

If the two do not exactly match, the user should terminate the data connection by issuing the ABOR command, and should not proceed to issue a data transfer command.

The format of the response code 225 reply is:

```
225 <SP> <text> <SP> (<d><proto><d><addr><d><port><d>) <CRLF>
```

The unformatted text portion of the reply (<text>) may be any message, and may be omitted. If present, it must be separated from the formatted portion of the reply by at least one space character.

The formatted portion of the reply must be enclosed in parentheses, formatted exactly as the arguments to the ESTP command, and has the same meaning.

4. Connection Management

Implementations should, for the purposes of connection management, treat the ESTA and ESTP commands as data transfer commands. As data transfer commands, the addition of the ESTA and ESTP commands does not significantly change connection management requirements.

Since these commands leave the connection established, but unused, a subsequent data transfer command should not cause establishment of a new data connection; the implementation should see that it has an established connection suitable for use with the desired data transfer and proceed to use that connection.

5. FEAT Response

The specifications for the ESTA and ESTP commands provide the means to reliably determine the Server-FTP process's support for the commands.

The user may issue a FEAT command [RFC2389] prior to use of the ESTA or ESTP command. This may allow the user to determine if the Server-FTP process supports these commands. However, since deployment of Server-FTP implementations which support the FEAT command is not yet wide-spread, the user should not rely upon implementation of the FEAT command to indicate implementation of the ESTA and ESTP commands.

A Server-FTP which implements the ESTA and ESTP commands may (from the point of view of this specification) implement the FEAT command. If so, it must include, in the response to the FEAT command, a feature line indicating support for the ESTA and ESTP commands.

The feature line must be formatted as:

<SP> ESTA <CRLF>

As specified in [RFC2389], the feature-name text, "ESTA," is not case sensitive, but should be transmitted in upper-case, and the feature line must begin with a single space character and end with a normal Telnet end-of-line sequence.

At this time, options are not envisioned for either command. To allow separate options for ESTA and ESTP, the user should also accept the text "ESTP" as indicating support for both commands. The user, however, should accept options on the FEAT response and discard any it does not recognize. When options appear, they must be separated from the feature-name text by exactly one space character.

6. IANA Considerations

The specifications in this memo make reference to assignment lists currently maintained by the Internet Assigned Numbers Authority (IANA); no new assignments are specified, and no new assignment lists are required.

The list of valid feature names sent in response to the FTP FEAT

command is believed to be first-come first-served, and managed outside the control of the IANA.

Specific examples taken from IANA-maintained lists at the time of this writing are for illustrative and informative purposes only.

7. Security Issues

The ESTP command, and the replies to both ESTA and ESTP, contain network addressing information which could be used to gather information about internal network architectures. However, since similar information is already transmitted on the FTP control connection, it presents no risk not already present in the FTP.

The intention of this protocol extension is to address the security issue involved with the problem of a race condition in the existing File Transfer Protocol [DS, CERT2]. This condition allows an attacker the ability to receive unauthorized transmissions from, or transmit unauthorized information to, the passive party of a file transfer. The attack makes use of the timing window between when the passive socket is prepared and when an active connection to that passive socket actually occurs.

The security industry identifier for this issue is CVE-1999-0351 [CVE].

Denial of Service attacks can make use of this issue. The described protocol is equally susceptible to Denial of Service attacks. It can, however, make their effect much more apparent since data transfers will fail where they may have incorrectly appeared to have succeeded without these protocol extensions.

The described protocol does not address the issue of "FTP Bounce" (CVE-1999-0017); where an attacker, via the control connection, instructs the Server-DTP to actively establish a data connection to a third party [CERT1].

The protocol presented also makes no attempt to secure the actual data transmission; it only provides a means to determine the proper data connection has been established prior to initiating data transfer over that channel. Alternatives, such as provided by the FTP Security Extensions [RFC2228, MURRAY], should be considered when data security is an issue. Unfortunately, these methods are not, as yet, widely deployed. ([MURRAY] is a work in process. Wide spread deployment, if it occurs at all, will be some time in the future.) The extensions provided by this protocol provide added measures which can compliment those alternatives, but cannot replace them.

The use of any Network Address Translation scheme, or certain proxy implementations, may render this protocol unusable. The protocol involves the transmission of network address information over the FTP control connection. Address translators, including proxy implementations which do not implement this protocol, can cause a mismatch between the information transmitted and the address actually observed at the communication channel end-point.

Acknowledgments

The following people provided significant assistance with the analysis of the problem, the proposed solution, and the preparation of this document:

The members of vulnerability handling team at the CERT Coordination Center.

Paul Ford-Hutchinson of IBM UK Ltd.

Ian Willis of Sun Microsystems.

Normative References

[RFC959] J. Postel and J. Reynolds, "FILE TRANSFER PROTOCOL (FTP)", STD 9, RFC 959, October 1985.

[RFC1123] IETF, "Requirements for Internet Hosts -- Application and Support", STD 3, RFC 1123, October 1989.

[RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirements Levels", RFC 2119, BCP 14, March 1997.

[RFC2389] P. Hethmon and R. Elz, "Feature negotiation mechanism for the File Transfer Protocol", RFC 2389, August 1998.

[RFC2428] M. Allman, S. Ostermann and C. Metz, "FTP Extensions for IPv6 and NATs", RFC 2428, September 1998.

Informative References

[RFC1639] D. Piscitello, "FTP Operation Over Big Address Records (FOOBAR)", RFC 1639, June 1994.

[RFC2228] M. Horowitz and S. Lunt, "FTP Security Extensions", RFC 2228, October 1997.

[MURRAY] P. Ford-Hutchinson, et al, "Securing FTP with TLS", IETF Internet-Draft, Work in process.

[POSIX] IEEE and The Open Group, "Base Specifications, Issue 6", IEEE Std 1003.1-2001, January 2001.

The following references are only available electronically. The URL given for each is believed to be the "original source." At least one is known to be no longer available from that source. All documents cited, however, are widely mirrored and known to be readily located using search engines.

[DS] D. Sacerdote, "Some problems with the File Transfer Protocol, a failure of common implementations, and suggestions for repair.", http://www.secnet.com/papers/ftp-paper.html, April 1996.

[CERT1] CERT Coordination Center, "Problems with the FTP PORT Command or Why You Don't Want Just Any PORT in a Storm", http://www.cert.org/tech_tips/ftp_port_attacks.html, April 1998.

[JG] J. Gerber, "FTP PASV "Pizza Thief" Exploit", http://www.infowar.com/iwftp/iw_sec/iw_sec_01.txt, February 1999.

[GL] G. Lundberg, "Security update for wu-ftpd 2.4.2 (beta 18) VR13", ftp://ftp.wu-ftpd.org/pub/wu-ftpd-attic/ANNOUNCE-2.4.2-beta-18-vr14, February 1999.

[CVE] Mitre Corp., "Common Vulnerabilities and Exposures", http://cve.mitre.org/, September 1999.

[KS] K. Seifried, "Problems with the FTP protocol", http://seifried.org/security/network/20010926-ftp-protocol.html, September 2001.

[CERT2] J. Lanza and J. Pickel, "File Transfer Protocol allows data connection hijacking via PASV mode race condition", http://www.kb.cert.org/vuls/id/2558, April 2002.

Author's Address

Gregory A. Lundberg
WU-FTPD Development Group
1441 Elmdale Drive
Dayton, OH 45409-1615
US

Phone: +1 937 299 7653
Email: lundberg@vr.net

Annex A - Address Family Numbers

The ESTP command, and the responses to both ESTA and ESTP, make use of the address family number. For informational purposes, the following table shows the assignments for the Internet Protocols as of the writing of this document. Refer to the IANA for a current list of all address family numbers.

AF Number	Protocol
1	Internet Protocol, Version 4 (STD 5, RFC 791)
2	Internet Protocol, Version 6 (RFC 2460)

Normative References for Address Family Numbers

Since publication of [RFC1700], the IANA has moved to an on-line database for management of the assigned numbers.

[IANA] IANA, "Protocol Numbers and Assignment Services", http://www.iana.org/numbers.html

[RFC1700] J. Reynolds and J Postel, "ASSIGNED NUMBERS", STD 2, RFC 1700, October 1994.

Annex B - Network Address Formats

The ESTP command, and the responses to both ESTA and ESTP, contain network addresses. Each network protocol has a required form for representation of its addresses. For informational purposes, the following table shows the formats for the Internet Procotols as of the writing of this document.

AF Number	Address Format	Example
1	dotted decimal	132.235.1.2
2	IPv6 string	1080::8:800:200C:417A

Normative References for Network Address Formats

[RFC952] K. Harrenstien, M. Stahl and E. Feinler, "DOD INTERNET HOST TABLE SPECIFICATION", RFC 952, October 1985.

[RFC2373] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture", RFC 2373, July 1998.

Annex C - Examples of Use

The following examples show typical uses of the ESTA and ESTP

commands detecting an unauthorized connection to the passive socket.

The assumption in these examples is that the TCP implementation on the host will establish multiple connections on a passive socket and return them to the implementation in the order established. Thus, a second active connection will succeed without error. Experience shows this assumption holds true for most existing implementations.

The examples all use the following IPv4 addresses, which are assumed routable on the example network without address translation.

user 192.168.1.2

The user; either actively connecting, passively accepting connections, or not participating in the data transfer.

active 172.16.3.4

The Server-FTP process actively connecting (PORT mode). The Server-FTP process will use TCP port 20, as assigned by IANA for this purpose.

passive 172.16.5.6

The Server-FTP process passively accepting connections (PASV mode).

attacker 10.7.8.9

The attacker actively connecting to the passive socket used in the particular example. The example attacker uses TCP port 20 in a simplistic (and likely successful) attempt to thwart fire-wall filtering rules, and to fool the passive FTP implementation (as well as intrusion detection systems which may be monitoring the traffic) into believing it is an actual Server-FTP implementation.

C.1 PORT Mode

The user prepares a passive socket and transmits its address to the active Server-DTP. The user's DTP is now listening for connections on that socket.

The window for the attacker to connect is now open.

user --> active: PORT 192,168,1,2,12,34

The attacker connects to the passive socket.

active --> user: 200 Okay.

The Server-DTP is now prepared to actively establish a connection to the specified passive socket.

user --> active: ESTA

The Server-DTP establishes the specified connection and returns the address of the active socket (the local address on the Server-DTP).

The window for the attacker to connect has now closed.

active --> user: 225 Connected from (|1|172.16.3.4|20|)

The user's DTP accepts the first connection from the passive socket. In this case, the connection from the attacker. It retrieves the active socket address (the remote address) and compares it to the information returned from the Server-FTP process.

Seeing a mismatch, the user immediately terminates the established data connection (to the attacker). Note the passive socket, and the established connection from the Server-DTP queued upon it, remain.

In most cases, however, since the passive socket is now "tainted," the user will desire to also close the passive socket. To do so, it must first instruct the Server-FTP process to abort the connection.

user --> active: ABOR

The Server-DTP closes its data connection.

active --> user: 226 Connection closed, abort successful.

The user may now close the passive socket and proceed with the remainder of its error-recovery process.

C.2 PASV Mode

The user requests the Server-DTP create a passive socket and report its address.

user --> passive: PASV

The Server-DTP prepares the requested passive socket. The Server-

DTP is now listening for connections on that socket.

The window for the attacker to connect is now open.

passive --> user: 227 Listening on 172,16,5,6,78,90

The attacker connects to the passive socket.

The user establishes the specified connection and sends the address of the active socket (the address local) to the Server-FTP process.

The window for the attacker to connect has now closed.

user --> passive: ESTP |1|192.168.1.2|43210|

The Server-DTP accepts the first connection from the passive socket. In this case, the connection is from the attacker. It retrieves the active socket address (the remote address) and returns it to the user.

passive --> user: 225 Connected to (|1|10.7.8.9|20|)

The user compares the address returned with the address assigned to its (local) active socket. Seeing a mismatch, it terminates the data connection prior to issuing a command which would actually transmit data.

user --> passive: ABOR

The Server-DTP closes the data connection (to the attacker). Note the passive socket remains, as does the user's connection queued upon it.

passive --> user: 226 Connection closed, abort successful.

The user proceeds with error recovery. In most cases, since the passive socket is now "tainted," the user will close its data connection and instruct the Server-FTP process to close the passive socket and prepare another.

C.3 Server-to-Server Mode

The user requests the Server-DTP create a passive socket and report its address.

user --> passive: PASV

The Server-DTP prepares the requested passive socket. The Server-DTP is now listening for connections on that socket.

The window for the attacker to connect is now open.

passive --> user: 227 Listening on 172,16,5,6,78,90

The user forwards the passive socket address to the second Server-FTP process.

user --> active: PORT 172,16,5,6,78,90

The attacker connects to the passive socket.

active --> user: 200 Okay.

The second Server-DTP is now prepared to actively establish a connection to the specified passive socket.

user --> active: ESTA

The second Server-DTP establishes the specified connection and returns the address of the active socket (the local address on the Server-DTP).

The window for the attacker to connect has now closed.

active --> user: 225 Connected from (|1|172.16.3.4|20|)

The user forwards this information to the first Server-FTP process, informing it to accept a passive connection.

user --> passive: ESTP |1|172.16.3.4|20|

The Server-DTP accepts the first connection from the passive socket. In this case, the connection is from the attacker. It retrieves the active socket address (the remote address) and returns it to the user.

passive --> user: 225 Connected to (|1|10.7.8.9|20|)

The user compares the address returned with the address provided by the second Server-FTP process. Seeing a mismatch, it proceeds to begin error-recovery by instructing the first Server-DTP to close the data connection (to the attacker).

user --> passive: ABOR

The first Server-DTP closes the data connection (to the attacker). Note the passive socket remains, as does the second Server-DTP's connection queued upon it.

passive --> user: 226 Connection closed, abort successful.

The user proceeds with error recovery. In most cases, since the passive socket is now "tainted," the user will want to instruct the first Server-FTP process to close the passive socket and prepare another. Before it does so, however, it must instruct the second Server-FTP process to close its data connection.

user --> active: ABOR

The second Server-DTP closes its data connection.

active --> user: 226 Connection closed, abort successful.

The user may now instruct the first Server-FTP process to close the passive socket and proceed with the remainder of the error-recovery process.

Annex D - Implementation Considerations

The ESTA and ESTP commands present some interesting possibilities for implementers; three of which are discussed here.

Experience shows the implementation of the TCP passive socket, on most hosts, queues incoming connections. While in the queue, the connections are established and simply not being used by the application. In [POSIX], for example, the accept function removes those connections from the queue, making them available to the application. This operation usually involves no network activity; to the remote end-point, the status of the established connection (queued or accepted) is transparent.

The formal specifications of the ESTA and ESTP commands indicate the implementation accepts the first connection queued upon the passive socket. There may, however, be several connections queued. One of them can be expected to be the intended connection.

D.1 Interactively Finding the Specified Connection

In sessions involving a passive Server-DTP, the user could make use of the (probable) behavior of TCP passive sockets in an attempt to locate the desired data connection.

Quite simply, after instructing the passive Server-FTP process to

close the data connection (to the attacker), instead of immediately requesting the preparation of a new passive socket, the user could issue another ESTP command.

Using this scheme, the user is instructing the passive Server-DTP to step through each connection queued on its passive socket until it (the user) finds one which is acceptable.

The danger is that attackers can establish new connections faster than the user can process them interactively. Thus, while this approach is viable, the user must guard against connection floods by imposing a limit (either by time or number of attempts) and proceeding on the basis of the last-accepted, mismatched connection once that limit has been reached.

A user, seeking maximum robustness, may implement this approach. The Server-FTP process should provide its own guards against connection flooding, either by using the automatic method outlined next, or by limiting (by time or attempts) the user's repeated use of the ESTP command.

D.2 Automatically Finding the Specified Connection

The implementer will note that, when using passive mode, the desired address is known prior to accepting a connection from the passive socket. For the user, this address was obtained from the response to the ESTA command. For the passive Server-DTP it was given with the ESTP command.

This presents the (rather appealing) possibility of stepping through each connection queued on the passive socket until one with the desired address is located.

It has the advantage of not involving any network traffic, as is required when interactively seeking the correct connection, and could more quickly reduce resources used (wasted) by the attacker connections.

Implementations should consider this approach. Since this is being done with local operations, it is significantly faster than the interactive approach. Connection floods, however, are still a possibility which must be guarded against.

D.3 Using a Common Passive Socket

An extension of automatically finding the desired connection is the possibility of using a common passive socket for all data transfers.

If the implementation issues with this approach can be addressed, it would greatly reduce the configuration issues faced by fire-wall administrators, and could significantly enhance both the usability and security of the FTP.

Basically, the Server-DTP always has a passive socket prepared, and reports the address of that socket in response to all PASV, LPSV and EPSV commands.

Those tempted to implement this approach are strongly cautioned to carefully consider all aspects, some of which are presented here. The discussion, however, is probably incomplete and bears careful analysis and further research by implementers.

The most obvious problem is that the implementation needs some means to queue connections already accepted, but not yet matched to sessions by the ESTP command. This could place a significant load on resources, especially when faced with a connection flood, and the total number of such connections possible may be restricted by host limitations.

Another problem is that some connections will never be matched. In terms of the passive race condition, they would represent attackers; but those connections could simply be the result of failed sessions. Queue-time limitations and other means might provide the solution to this problem.

The appeal of this possibility, however, is so great that it bears further research. If the implementation issues can be properly addressed, whether through protocol enhancements or through implementation guidelines, this could become the preferred mode of use for passive sockets in Server-FTP implementations.

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by The Internet Society.