FTP/TLS Friendly Firewalls

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html
Index

1. .......... Abstract
2. .......... Introduction
3. .......... Audience
4. .......... Problem Statement
5. .......... Technical Description
5.1. .......... Establishing a Data Connection
5.2. .......... Data connection behaviour for FTP
5.3. .......... Port numbers
6. .......... Firewalls
6.1. .......... Port Filtering Firewalls
6.2. .......... Content Aware Firewalls
6.3. .......... Network Address Translators
6.4. .......... Application Layer Gateways
7. .......... Trying to secure FTP with TLS over firewalls
7.1. .......... Port Filtering Firewalls
7.2. .......... Content Aware Firewalls
7.3. .......... Network Address Translators
7.4. .......... Application Layer Gateways
8. .......... Premature Control Termination
9. .......... midcom
10. .......... Security Considerations
11. .......... IANA Considerations
12. .......... Network Management
13. .......... Internationalization
14. .......... Scalability & Limits
15. .......... Acknowledgements
16. .......... References
17. .......... Authors’ Contact Address

Appendix

A. .......... Firewall rule summary
1. Abstract

This document discusses some of the issues with running FTP, secured with TLS as defined in [FTP-TLS], through firewalls. FTP is known to be a bit of a problem for firewalls (see [RFC-1579] for a discussion of normal FTP and firewalls). Some of the problems have been fixed by adding intelligence into the firewall. With secured FTP, where the control connection is encrypted, some of these techniques fail.

Whilst this document confines itself to issues of FTP over TLS, the issues will probably be relevant for most secured FTP protocols that conform to [RFC-2228]. Some of the discussions will also be relevant to any protocol that firewalls do clever things with.

2. Introduction

This document is presented as a discussion of secure FTP behaviour as viewed by a firewall. It discusses some of the techniques that firewalls use to provide a path for the FTP protocol and how those are affected by layering FTP on TLS.

3. Audience

This document is aimed at designers of firewall software and people involved in deploying firewalls in an environment where FTP over TLS needs to traverse them.

4. Problem Statement

The FTP protocol has two troublesome features that cause headaches when trying to pass over boundary firewall devices.

- There are two distinct connections used
- The information used to establish the second connection is dynamically created and passed between the entities over the first connection.

The two connections are the Control Connection and the Data Connection. In practice, there is usually one Control Connection per session and numerous Data Connections. The operation of FTP defines that the following data transfers take place on a Data Connection:

- File sending, using the STOR command
- File retrieval, using the RETR command
- Listings, using the NLST and LIST commands (and the MLST command)

Implementations usually establish a data connection for each of the data transfers. The protocol documentation [RFC-959] does not specify this behaviour, but does allow for it.

A note about Transmission Modes: FTP [RFC-959] defines three modes of data transfer, Stream, Block and Compressed. In practice Stream mode, the only one required by [RFC-1123] and the default mode, is the most widely used. [RFC-1123] states that clients that use Stream mode SHOULD use a PORT command. Stream mode data transfers are ended by closing the data connection, effectively forcing one data connection per data transfer.

5. Technical Description

For the discussions below, assume that a Client has connected from port ‘U’ to the FTP port ‘L’ on the Server. (The well known port for FTP is ‘21’, thus, normally, ‘L’ is ‘21’). Also assume that the transmission mode is ‘Stream’.

5.1. Establishing a Data Connection

A Data Connection needs to be established for the data transfer commands (STOR, RETR, NLST, LIST and MLST) to operate. When the server receives one of these commands, it replies with an intermediate reply (‘150’) indicating that it is ready to transfer the data. The Data Connection is established and the data transfer happens.

5.2. Data connection behaviour for FTP

If no PORT or PASV command is issued, the Client should listen for a connection from the Server on the same port as the control connection (‘U’). The Server should initiate the Data Connection from the default Data Port, which is ‘L - 1’ (‘20’).

If a PORT command is to be issued then the Client should obtain a port number and pass that to the Server in the PORT command (as a comma separated list "h1,h2,h3,h4,p1,p2" where h1 is the high order 8 bits of the internet host address.) The Server should then establish a connection to the Client on the Address and Port indicated, from the Server’s Default Data Port (‘L - 1’). Note: This does not need to be on the host at the other end of the Control Connection.
If the Client wishes to use Firewall Friendly FTP [RFC-1579] then the Client issues a PASV command, which causes the Server to listen on a port (not the default data port). The Server indicates which port it is listening on, in the 227 response to the PASV command (see [RFC-1123]). This time the Data Connection is initiated from a port on the Client to the port indicated by the Server. Note: This does not need to be on the host at the other end of the Control Connection.

5.3. Port numbers

The TCP and UDP addressing mechanism has two parts. A Host Address and a Port number. In IPv4 a host address is a long integer (32 bits) and a port number is a short integer (16 bits). Each host may use those port numbers to originate or listen to connections. At a ‘C’ coding level, there are three basic ways to get a port for a connection.

- get a socket and then connect to a remote host
- get a socket, bind to port 0 and then query the socket information to find out which port number you were given.
- get a socket and bind to a specific port number.

The first mechanism is only really useful for obtaining a port to originate a connection; the second two mechanisms are suitable for obtaining a port for originating or for listening for a connection.

There are three ranges of port numbers:

- Well known ports. These are the port numbers less than 1024. On Unix systems, it is usually a restriction that these ports are only accessible to a process running with ‘root’ privilege. It was therefore assumed that services running in this port range would be system services, running with the consent of the system administrator, and therefore trustable. With the advent of home linux systems and Windows systems that do not have a ‘root’ user - this basis of ‘trust’ is not reliable. Widely used Internet protocols will usually be on ports <1024, user processes will not normally (even on Windows machines) be allocated ports under 1024 unless specifically requested.

- Registered ports. These are the port numbers in the range 1024-49151. These are for services that do not need to be run with root privilege, but do need to have a port number agreed by both the client and server.
6. Firewalls

Simple port filters rely on the ‘well known port’ system that underpins much of the establishment of TCP (and UDP) based protocol conversations. Each service has a well known port and a client and server should expect to find each other on those ports. Examples of the common ports are: 80 for http; 443 for https; 25 for SMTP; 23 for telnet and 21 for FTP. Such firewalls boldly assume that machines on either side will be well behaved and only offer the correct service on the correct port. In this way, they are able to filter unwanted traffic between hosts by examining the source address/port and destination address/port and deciding if the hosts are allowed to use the service indicated.

For the FTP protocol, two rules must be set up in the Firewall.

The first rule allows the client to connect to the Server for the Control Connection.

The second rule is for the Data Connection and will depend if Normal or Firewall Friendly FTP is to be used (or both).

6.1. Port Filtering Firewalls

The problem that simply using port filtering for FTP generates is that the data connection rule tends to open up quite a large hole in the Firewall and many implementors do not wish to define it. The more general problem with simple port filtering is the issue of port number misuse. To fix both of these issues, a Content Aware firewall may be deployed.

6.2. Content Aware Firewalls

In addition to port filtering rules, Content Aware firewalls will also look at the content of the conversation and may perform actions based on what they observe. This has two impacts for the FTP protocol. Firstly, it allows the firewall to observe the content of the Control Connection and make decisions based upon it. Secondly, it allows the firewall to open up temporary holes for the Data Connection, based upon the content of the PORT command and/or PASV response.

Content Aware firewalls:
- may only allow certain FTP commands to be used

- may have restrictions in the parameters to certain commands
  (e.g. STOR, RETR, CWD, NLST and LIST preventing any directories
   with a certain name being accessed)

- may insist that all commands conform to some expected criteria,
  such as being ended by CRLF delimiters.

When a Content Aware firewall observes a PORT command or the 227
reply to the PASV command, it may dynamically create a rule to
allow the indicated Data Connection to pass through. This removes
the requirement for the quite wide definitions that would
otherwise be needed to allow the data connection to be
established.

6.3. Network Address Translators

One step up from Content Aware firewalls are boundary devices
which allow addresses to be hidden from the outside world. There
are two major reasons for this. The first is to hide any network
topology information; the second is to allow the use of private
network addresses (see [RFC-1918]). For the FTP protocol, Network
Address Translators need to read and modify PORT commands and/or
PASV responses to substitute their own address and port for those
indicated on the Control Connection.

6.4. Application Layer Gateways

These types of boundary devices actually understand the FTP
protocol and act as a application layer proxy between the two
hosts. To the Client it acts as a server, to the Server it acts
as a client.

7. Trying to secure FTP with TLS over firewalls

If we look at our four category of devices, we can examine the effect
of deploying [FTP-SSL] secured FTP over TLS.

7.1. Port Filtering Firewalls

FTP over TLS will not affect the operation of the FTP protocol as
viewed by a simple port filtering firewall. The connections and
the ports are exactly the same

7.2. Content Aware Firewalls

Content aware firewalls will no longer be able to understand the
content of the Control connection. This means that

- Any packet on the control connection that cannot be inspected, must be configured to be passed through.

- Normal port filtering firewall rules must be in place for the Data Connection, as the firewall will not be able to open up the pinholes, based on examination of the Control Connection.

7.3. Network Address Translators

NAT firewalls will not work for secure FTP if the NAT will affect the PORT address or the PASV response address. An FTP Client on a NATted network should be able to use secure FTP over TLS in firewall friendly mode to a Server that has a real Internet IP address. An FTP Client with a real Internet IP address should be able to use an FTP Server that is on a NATted network in normal mode (assuming there is some mechanism for the Client to establish a Control Connection).

7.4. Application Layer Gateways

In general Client-Server secured FTP will not work at all with an ALG. However, It may be possible to configure an ALG as one of the endpoints of the secured FTP session as it flows over a hostile network.

8. Premature Control Termination

Another issue with firewalls and FTP is connected to the problem of timeouts. Due to the two-connection model of FTP, there is a high likelihood that the Control connection will have no activity during a data transfer. In the case that the data transfer is long, the firewall may incorrectly assume that the Control connection is no longer needed and close it down. Thus, the data transfer will complete correctly, but the 226 reply on the control connection will not be received and the client and server will, eventually, time out independently.

9. midcom

One approach to help solve the issue is the MiddleBox communications working group. Their aim is to create a model and set of protocols to define a communications protocol between endpoints and boundary devices, such as firewalls, that will allow the client or server to request a path through the firewall without the firewall itself needing to be able to understand the protocol that it is passing
10. Security Considerations

This document attempts to explain how the FTP protocol operates from a firewall’s perspective; how a firewall can be configured to allow FTP (and secure FTP) to traverse it and how some of the more advanced features of firewalls and application layer gateways can make life hard for secured protocols.

11. IANA Considerations

(FTP-PORT) - The port assigned to the FTP control connection is 21.

12. Network Management

NONE

13. Internationalization

NONE

14. Scalability & Limits

NONE

15. Acknowledgements
16. References


          RFC 2228, October 1997.


          RFC 2577, May 1999.

[RFC-2817] R. Khare, S. Lawrence, "Upgrading to TLS Within HTTP/1.1"

[RFC-2818] E. Rescorla, "HTTP Over TLS"

[FTP-EXT] R Elz, P Hethmon "Extensions to FTP"

[FTP-TLS] "Securing FTP with TLS"
17. Authors’ Contact Address

The FTP-TLS draft information site is at http://www.ford-hutchinson.com/~fh-l-pfh/ftps-ext.html

Please send comments to Paul Ford-Hutchinson at the address below

Paul Ford-Hutchinson
IBM UK Ltd
PO Box 31
Birmingham Road
Warwick
United Kingdom
tel – +44 1926 462005
fax – +44 1926 496482
email – paulfordh@uk.ibm.com
A. Firewall rule summary

As long Application Layer Gateways (or proxys) are not used, a packet filtering firewall should be able to pass secured FTP. The following guidelines should help trying to configure one.

Control Connection

- Allow any port on the client to connect to port 21 on the server
- Disable any rules that parse and/or impose any rules on the commands and/or responses on the control stream.
- Ensure the idle timeout of the control connection is longer than it will take to transfer the largest file on the data connection

Data Connection

Normal (active or PORT) FTP

- Allow port 20 on the server to connect to any port on the client

Firewall-Friendly (passive or PASV) FTP

- Allow any port on the client to connect to any high port(*) on the server.

  (*) This may be able to be configured on the server to be a range of ports and not ‘any high port’.

Note: A firewall may allow both Normal and Firewall-Friendly FTP, the choice is not exclusive.

NAT firewalls should be able to allow Firewall friendly FTP through, as long as these rules can be followed.