

A Comparison of Secure File Transfer Mechanisms

Introduction

In the interest of protecting customer data or securing trade secrets many companies are modifying their mechanisms of transferring data across the Internet. There are a number of things to consider when improving the security of data transfer procedures, these include:

- User Authentication – FTP has traditionally used clear text passwords. This weakens the security as someone can pick up the password that is used and use it later to get access to the data.
- Remote system identification verification to prevent hijacking of the packets by a system masquerading as the destination.
- Data privacy (encryption) so that no intermediate system can use the data.
- Data security (integrity or tamper prevention) to prevent modification to the data while it is in transit.
- Preservation of data format. Different operating systems may store data in different formats. It is desirable to have a defined interchange format when this is the case. FTP has traditionally done a very good job of performing data interchange between systems.
- Ease of use. A mechanism that requires extra steps or is not easy to use will encourage users to take short cuts that may not preserve the desired security when they are in a hurry.

A variety of mechanisms are discussed below. These mechanisms are: separate encryption, SFTP (Secure Shell File Transfer Program), FTP over Secure Shell (SSH), IPSec, Virtual Private Networks, and FTP over Transport Layer Security (TLS). Each mechanism has arguments for and against it so no one can be declared the solution to all problems.

Separate Encryption before using FTP

Encryption of the data by a separate program before performing the transfer was probably the first method used to solve this problem. Though this method is readily available, it doesn't solve all of the problems. This method doesn't protect the user's password, so someone spying on the transmission could get access to the data after it has been decrypted unless a separate mechanism is used to limit the reusability of passwords.

The requirement for manual encryption could cause problems when the user is in a hurry or discovers that there is a file that is needed that wasn't encrypted before the transfer session was started. When the source

and destination systems run different operating systems, it is possible that the data also needs to be converted either before or after it is encrypted. While this may be taken care of by the encryption program it is something to take in account when evaluating this method. Separate encryption may not provide data integrity. There are no mechanisms for the server to certify that it is the intended system in this method. This is the weakest method.

SFTP (SSH File Transfer Program)

SFTP is widely available for a number of platforms and it solves the problems of securing the user's password and provides data encryption and integrity on the fly. SSH (which SFTP uses as an authentication and data transport mechanism) also authenticates the server involved through the exchange of keys. SSH keys are privately maintained and require external acceptance upon first use or prior transfer through an alternate method.

Since SFTP only uses a single TCP connection to exchange both commands and data it does not have the problems with firewalls that FTP can have. Unfortunately SFTP doesn't always preserve the file format when different operating systems are involved.

The SFTP protocol was originally specified as a binary file access protocol. Though a text transfer mechanism was added in later revisions of the specification, not all implementations support it, particularly the most popular (OpenSSH). The protocol also provides a mechanism for arbitrary file characteristics to be passed on the file open command, but this is not highly used. The SSH community has allowed the draft specification to expire as it was felt that the group didn't have the necessary expertise to standardize a file access protocol. While there are many implementations available, not all of them will update the protocol level that they support if they don't feel that their market has a use for it. This method provides acceptable functionality for many users though the user may have to experiment to deal with text file format conversion issues.

FTP over SSH

SSH can be used to create a secure tunnel between two systems. It is possible to have one end of this tunnel point to an FTP server and provide a secure channel for FTP transfers. Some SSH servers and clients recognize the FTP PORT and PASV commands and replies and can provide protection for the data channel as well. To use this method an SSH connection must be established between the two systems before the FTP connection is established, which adds inconvenience or uses resources even when there are no transfers being done. With this method SSH provides data privacy and integrity, server identification verification and privacy for the user password. FTP provides any data format conversion that is necessary between the two systems.

IPSec (and FTP)

IP Security (IPSec) provides secure communications (authentication, integrity, confidentiality) over IP-based networks between systems. Not all systems have IPSec available. Even when systems have it available, configuring differing types of systems to work together can be a challenge. Since this needs to be configured on a per system basis it may lack flexibility when destinations or sources change frequently.

Since IPSec protects the individual packets sent between nodes it can present a problem if one of the nodes is operating behind a NAT device that does not support IPSec NAT traversal (RFC Process Software Secure File Transfer Mechanisms 3947, 3948). Depending upon how it is configured IPSec can provide both data integrity and data security. If data protection is desired, then it is necessary to configure IPSec to encrypt all traffic between the two systems as FTP may use an arbitrary data port for the transfer. Configuring IPSec to encrypt all data transferred between two addresses may expend CPU cycles where it isn't desired. Since IPSec is implemented in the lower layers of the IP protocol use of it can lead to a high amount of CPU cycles being used at interrupt or kernel levels. IPSec generally requires configuration by a system administrator and therefore users may be delayed in performing transfers while waiting for new configurations to be entered and tested. IPSec provides host authentication while doing key negotiation and during data transfer.

Virtual Private Networks (and FTP)

A Virtual Private Network uses encryption to provide secure communication between two systems. It may do it at network layer 2, by creating a logical wire between the two systems. In this case all network traffic passes over this logical wire, whether or not is destined for the system on the other end. Or it may create it at network layer 3 by encrypting and encapsulating packets that traverse a particular route. A VPN can also be created with external (router) hardware being configured to encrypt the data between specified addresses. Not all systems support the creation of a virtual private network (VPN), and not all systems that do offer compatible mechanisms. The VPN setup mechanism may or may not provide a method of verifying the identity of the remote system. A VPN may end up carrying (and encrypting) data that does not need to be carried over a secure network connection if it operates at layer 2, or if routing characteristics direct all traffic through it. This method also has the potential problem of limited adaptability to quick changes.

FTP over TLS (FTPS)

Transport Layer Security (TLS, RFC 2246) is commonly used to secure data transferred between web browsers and servers (https). TLS is also known as Secure Socket Layer (SSL). FTP over TLS is specified by RFC 4217 and uses TLS to add password privacy and server verification to FTP. It also makes privacy for data transfers available. The command channel is protected during the user authentication procedure and may be set to clear after setting file transfer protection requirements to let firewalls and NAT devices learn about the FTP data channel and open the necessary ports to allow the data to be exchanged. The data channel may be set to private, which provides both data security and integrity. FTP was designed to properly handle ASCII and binary file transfers so it does well when different system types are involved. The data encryption takes place after the data is converted to standard interchange formats so conversion of file formats is not an issue. Since the data protection features are integrated into FTP the CPU cycles

necessary to provide it are expended at user level. This functionality is available for many systems, but may not be available for all as it requires integration with the FTP utility. TLS provides server authentication with keys that may be either self-signed or signed by a trusted authority. Servers may be configurable to require secure data transfers. FTP over TLS requires an explicit request for encryption and server authentication so use of the secure channel can be optional.

Summary

There are a number of methods available for providing security to data transfers. It is hard to say that a particular one is the best, as functionality and availability of implementations are a key concern. A system that provides a variety of methods has a higher possibility of allowing data to be exchanged in a usable format with necessary security and a minimum amount of inconvenience.

Method	Password Privacy	Server Verification	Data Privacy	Data Integrity	Text and Binary Transfers	Automatic
Separate Encryption	No	No	Yes	Maybe	Maybe	No, file must be encrypted and decrypted separate from the transfer.
SFTP	Yes	Yes, private key	Yes	Yes	Maybe	Yes
FTP over SSH	Yes	Yes, private key	Yes	Yes	Yes	No, separate setup of the SSH tunnel is needed.
IPSec	Yes	Yes, private key or certificate authority	Yes	Yes	Yes	Yes, but configuration can be a challenge.
VPN	Yes	Maybe	Yes	Yes	Yes	Maybe, but may have configuration challenges.
FTPS	Yes	Yes, private key or certificate authority	Yes	Yes	Yes	No, but is easy to use.

About Process Software

Process Software has been a premier supplier of communications software solutions to mission critical environments for twenty years. We were early innovators of email software and anti-spam technology. Process Software has a proven track record of success with thousands of customers, including many Global 2000 and Fortune 1000 companies.



U.S.A.: (800) 722-7770 • International: (508) 879-6994 • Fax: (508) 879-0042

E-mail: info@process.com • Web: <http://www.process.com/>