Interested in learning
more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

Securing FTP Authentication

The File Transfer Protocol, or FTP, is an industry standard method of data exchange between computers. Widely used because of its flexibility and ubiquity, FTP has also become a frequent point of attack. Though certainly not the only issue, one frequently cited area of concern is the use of a clear-text data stream for passing authentication and control information. Intended for a novice to intermediate level administrator, this paper briefly examines how a nonsecure FTP implementation functions and demonstrates how th...

Copyright SANS Institute
Author Retains Full Rights

AD

Securing FTP Authentication
Mike Gromek
February 12, 2002

Summary

The File Transfer Protocol, or FTP, is an industry standard method of data exchange between computers. Widely used because of its flexibility and ubiquity, FTP has also become a frequent point of attack. Though certainly not the only issue, one frequently cited area of concern is the use of a clear-text data stream for passing authentication and control information.

Intended for a novice to intermediate level administrator, this paper briefly examines how a non-secure FTP implementation functions and demonstrates how the clear-text control connection can be exploited. A common misconception is that switched network architectures adequately protect an organization from network eavesdropping. Several ways of bypassing switch security are outlined, illustrating the continuing need for protecting the FTP data streams.

Having recognized this as an ongoing problem, the Internet community has drafted a series of FTP security extensions, providing a mechanism to establish a secure connection. These extensions are discussed and several more secure FTP implementations are briefly examined, illustrating different approaches to this problem.

How does FTP work?

A full description of how FTP works is well beyond the scope of this paper, however a brief overview is appropriate. The full description of a standard FTP implementation is available in IETF Standard 9, RFC-959 [1].

In a standard FTP implementation, an FTP server listens for network connections on the designated port (TCP port 21, by convention). A client then establishes a TCP socket connection to the server on this port. This connection is known as the "control connection" or "control channel", and is used to pass FTP commands and responses between the client and server.

Once the control channel is established, the server sends a message indicating the session has been established and that it is ready for a user to login. The client then sends a user name and password (if required) to complete the authentication. (FTP servers can be configured to accept anonymous logins, requiring no password.)

Once authenticated, a number of commands are available that allow the user to move around the file system, indicate what type of file transfer is preferred and perform a number of other useful tasks. At some point, however, the user will most likely initiate a file transfer. The FTP model actually supports three different ways of transferring a file, however all employ the use of a second network connection, the "data connection" or "data channel".

Three types of data connections can be established:

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

1. **Server-initiated connection.** Under this model, the server opens a connection to the client, using a different local port number (TCP port 20, by convention) and the same client port as the control connection. This approach limits the number of ports required on the client side. Alternatively, the client can provide a second, distinct port number for the data connection. In either case, the data connection is initiated from the server to the client, a process that can be problematic in firewalled environment. (Most firewalls are configured such that they will not allow external connections to client machines.)

2. **Client-initiated connection.** To avoid problems with firewalls, the client can use the PASV command, asking the server to open a second port for the data connection. The server obtains a port and returns both its IP address and the port number to the client. The client then opens the data connection using this second port number. Because the client initiates the connection, this is more easily handled at the firewall. (See RFC-1579 for additional details. [2])

3. **Proxy connection.** A third type of transfer exists, where the client actually initiates the data transfer via a second server. Essentially, the client requests that the first server (Server A) listen on a port (similar to the client-initiated connection described above). The client then passes Server A's IP address and the port number to Server B, which establishes a connection to Server A. The client then initiates the data transfer, requesting one server to send data and the other to receive it. This scenario is useful, for example, where an organization does not want client machines to have direct access to the Internet. The organization can provide a secure, hardened server to act on behalf of the client machine.

A more complete description of each of the above models can be found in the Internet Draft "Securing FTP with TLS" [3].

Why is FTP considered a security risk?

There are several reasons why FTP is considered problematic in a secured environment.

- **Programming errors.** A quick search of the CERT website (http://www.cert.org/advisories) lists multiple advisories resulting from bugs in various versions of FTP. The most recent advisory was released in November 2001 and, as of this writing, vendors are still just responding with fixes [4].

- **Misconfiguration.** Many FTP servers do not control or limit access to underlying operating system. Frequently, a person able to make an FTP connection can copy, and in some cases replace, critical system files. Constructing a properly secured FTP server is a significant undertaking and interested readers are referred to material dealing specifically with this subject [5] [6] [7].

- **"Bounce" attacks.** Using a proxy connection, an attacker can use an intermediary, authorized server to place data on an unauthorized machine. The data is effectively "bounced" through the authorized machine to the ultimate target. [8]

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

- **Clear-text authentication.** The standard FTP control channel is ASCII-based and not encrypted. As a result, it is often a trivial exercise to acquire a username and password.

This paper focuses on the final concern, clear-text authentication.

How easy is it, really?

Given the right network access, actually eavesdropping on an FTP session is trivial and can be done using readily available tools. The following example will illustrate how to capture an FTP session using tcpdump. Tcpdump ships with most versions of Unix, including most distributions of Linux. Versions can be downloaded for most Windows platforms and the few versions of Unix that do not include it by default. Tcpdump is available from http://www.tcpdump.org.

The following is an actual typescript of a sniffed FTP session. The IP addresses and hostnames have been sanitized, but this transcript is otherwise unaltered. Buried in the data stream, we can see the exchange of credentials (highlighted in bold for legibility). In this example, the user "ftpuser" is connecting to the server "ftpserver".

```
Script started on Mon Jan 21 18:38:00 2002
[root@sah /root]# tcpdump -i eth0 -a tcp port 21
Kernel filter, protocol ALL, TURBO mode (575 frames), datagram packet
socket
tcpdump: listening on eth0
18:38:36.885350 P client.example.org.3940 > 192.168.152.4.ftp: S
309703138:309703138(0) win 61440 <mss 1460>

                E^@ ^@ ,  }.. ^@^@  <^F  t^L ....  s |
               .... ..^D ^O d ^@^U ^R u .... ^@^@ ^@^@
                `^B ..^@  G.. ^@^@ ^B^D ^E.. ^@^@
18:38:36.895350 P 192.168.152.4.ftp > client.example.org.3940: S
2782268535:2782268535(0) ack 309703139 win 24820 <mss 1460> (DF)

                E^@ ^@ , ..^W  @^@  ;^F ^S.. .... ..^D
               ....  s | ^@^U ^O d .... ^L w ^R u ....
                `^R  `..  $ 1 ^@^@ ^B^D ^E..  U U
18:38:36.895350 P client.example.org.3940 > 192.168.152.4.ftp: . 1:1(0)
ack 1 win 61320 (DF)

                E^@ ^@ (  }..  @^@  <^F  4^O ....  s |
               .... ..^D ^O d ^@^U ^R u .... .... ^L x
                P^P .... .. Y ^@^@ ^@^@ ^@^@ ^@^@
18:38:36.915350 P 192.168.152.4.ftp > client.example.org.3940: P 1:46(45)
ack 1 win 24820 (DF)

                E^@ ^@ U ..^X  @^@  ;^F ^S.. .... ..^D
               ....  s | ^@^U ^O d .... ^L x ^R u ....
                P^X  `.. ^O^K ^@^@  2 2  0   f t  p s
                e r  v e  r   F T  P   s e  r v  e r
                  (  S u  n O  S   5 .  8 )   r  e a
                d y  .^M ^J
```

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

```
18:38:36.915350 P client.example.org.3940 > 192.168.152.4.ftp: . 1:1(0)
ack 46 win 61275 (DF)

                E^@ ^@ (  }..  @^@  <^F  4^N ....  s |
               .... ..^D ^O d ^@^U ^R u .... .... ^L..
                P^P .. [ .. Y ^@^@ ^@^@ ^@^@ ^@^@
18:38:41.085350 P client.example.org.3940 > 192.168.152.4.ftp: P 1:15(14)
ack 46 win 61320 (DF)

                E^@ ^@ 6  }..  @^@  <^F  3.. ....  s |
               .... ..^D ^O d ^@^U ^R u .... .... ^L..
                P^X .... .... ^@^@  U S  E R   f  t p
                u s  e r ^M^J
18:38:41.085350 P 192.168.152.4.ftp > client.example.org.3940: . 46:46(0)
ack 15 win 24820 (DF)

                E^@ ^@ ( ..^Y  @^@  ;^F ^S.. .... ..^D
               ....  s | ^@^U ^O d .... ^L.. ^R u ....
                P^P  `..  ;.. ^@^@  U U  U U  U U
18:38:41.085350 P 192.168.152.4.ftp > client.example.org.3940: P 46:82(36)
ack 15 win 24820 (DF)

                E^@ ^@ L ..^Z  @^@  ;^F ^S.. .... ..^D
               ....  s | ^@^U ^O d .... ^L.. ^R u ....
                P^X  `.. .... ^@^@  3 3  1   P a  s s
                w o  r d   r  e q  u i  r e  d   f o
                r   f t  p u  s e  r . ^M^J
18:38:41.085350 P client.example.org.3940 > 192.168.152.4.ftp: . 15:15(0)
ack 82 win 61284 (DF)

                E^@ ^@ (  }..  @^@  <^F  4^L ....  s |
               .... ..^D ^O d ^@^U ^R u .... .... ^L..
                P^P .. d ..^^ ^@^@ ^@^@ ^@^@ ^@^@
18:38:43.925350 P client.example.org.3940 > 192.168.152.4.ftp: P 15:28(13)
ack 82 win 61320 (DF)

                E^@ ^@ 5  }..  @^@  <^F  3.. ....  s |
               .... ..^D ^O d ^@^U ^R u .... .... ^L..
                P^X .... ..^Y ^@^@  P A  S S   a  b c
                d e  f^M ^J
18:38:43.935350 P 192.168.152.4.ftp > client.example.org.3940: P
82:111(29) ack 28 win 24820 (DF)

                E^@ ^@ E ..^[  @^@  ;^F ^S.. .... ..^D
               ....  s | ^@^U ^O d .... ^L.. ^R u ....
                P^X  `.. .... ^@^@  2 3  0   U s  e r
                  f  t p  u s  e r   l  o g  g e  d
                i n  .^M ^J
```

In fact, it is not actually necessary to scan through all of the confusing tcpdump output to find information of interest. In the following example, we write the tcpdump output to a binary file and simply scan for strings. Along with the authentication information we are looking for (and a bit of noise), we find other interesting information, such as the operating system version (Solaris 8, in this case) and a complete record of the user's activity on the system.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

```
[root@sah /root]# tcpdump -i eth0 -w ftp.out -a tcp port 21
Kernel filter, protocol ALL, TURBO mode (575 frames), datagram packet
socket
tcpdump: listening on eth0
^C
33 packets received by filter
[root@sah /root]# strings ftp.out
220 ftpserver FTP server (SunOS 5.8) ready.
(~ @
USER ftpuser
UUUUUU
331 Password required for ftpuser.
PASS abcdef
230 User ftpuser logged in.
SYST
215 UNIX Type: L8 Version: SUNOS
257 "/tmp" is current directory.
TYPE I
200 Type set to I.
PORT 192,168,115,124,15,103
200 PORT command successful.
RETR /etc/passwd
150 Binary data connection for /etc/passwd (192.168.115.124,3943) (508
bytes).
226 Binary Transfer complete.
L<v8
QUIT
L<v8
221 Goodbye.
L<v8
L<v8
UUUUUU
L<v8
UUUUUU
L<v8
L<v8
UUUUUU
[root@sah /root]# ^D

Script done on Mon Jan 21 18:40:35 2002
```

What is "the right network access"?

We prefaced the previous section by saying "Given the right network access…". Network sniffers, such as tcpdump, work by putting a machine's Ethernet adapter into "promiscuous" mode. Normally, an Ethernet card will only accept packets that have its local MAC address as the destination address. In promiscuous mode, the card will accept all packets sent on the Ethernet segment, regardless of the destination address.

In other words, in order to sniff an FTP session as illustrated above, the attacking machine must be able to see traffic sent between the client and the server. This is the case, for example, when the attacking machine shares the same Ethernet hub as either the client or the server.

© SANS Institute 2002, Author retains full rights.

But my network uses switches, not hubs…

As more and more shared hubs are replaced with network switches, a common misconception is that switched network architectures provide protection against network eavesdropping.

There are a number of techniques that can be used to sniff traffic in a switched environment. For example, an attacker can exploit any of the following:

- **Workgroup or "cubicle" hubs.** Although shared media is being displaced at the network core, it is not uncommon to see hubs used to provide additional connectivity in offices and cubicles. After all, it's cheaper and faster to grab one the unused hubs from the data center than it is to run additional network drops to an office. Usually, these devices are plugged in under a desk and never looked at again. (Or, at least, not until someone kicks the power cord out, six months down the road.) And, more often than not, the people requiring the additional connectivity are the technical staff: system administrators, DBAs and programmers - exactly the people an attacker wants to target.

- **Trunk and SPAN ports.** Most switches provide a trunk port. This port receives all traffic on the switch and is normally used to connect two switches. Access to the trunk port generally requires physical access to the switch. Some vendors, however, allow any port to be configured to receive traffic destined for another port. An example of this is a SPAN port on a Cisco switch. Since a SPAN port is configured in software, an attacker gaining remote administrative access to a switch can easily allow a desktop machine to sniff any traffic without physical access to the device.

- **ARP-cache poisoning.** Although more difficult to perform, tools such as arpspoof [9] and parasite [10] can be used to corrupt a victim's ARP table. This attack allows an attacking machine to masquerade as a legitimate device, often a router or firewall. The attacking machine then receives the data stream, views or manipulates it as desired, and passes it on to the original target.

- **DNS poisoning.** Similar to ARP-cache poisoning, this attack corrupts a DNS server or cache such that the attacking machine's IP address is returned in place of the original target. Again, the data stream is intercepted, viewed and relayed to the original recipient.

- **ICMP Redirects.** Again, traffic is redirected to an attacking machine, this time by issuing an ICMP Type 5 ("Redirect Error") message. This message informs a router that a better route exists - through the attacking machine.

- **MAC Flooding.** Switches must keep track of which computer is on each port. On many switches, flooding the switch with invalid MAC addresses will effectively exhaust its capacity to track which computer is where. Once this occurs, the switch reverts to broadcasting traffic, effectively turning the device into a hub.

- **MAC Duplication.** Since switches forward traffic based on MAC addresses, an attacker can impersonate a victim simply by presenting the same MAC address to the switch. In some cases, the switch is unable to manage having the same MAC address on two

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

different ports and will simply fail. In many cases, however, the switch will happily forward the traffic to both machines simultaneously.

A more complete discussion of some of these techniques can be found in SANS paper "Why your switched network isn't secure". [11]

Solving the problem

Not surprisingly, the Internet community has recognized the weakness of clear-text authentication and has amended the FTP specification to allow the use of encryption. RFC-2228 ("FTP Security Extensions") defines eight additional FTP commands to be used to negotiate an encrypted connection between the client and server. [12]

RFC-2228 allows a client to optionally issue the AUTH command in place of the standard USER, PASS or ACCT commands. The AUTH command passes a string indicating the security mechanism the client wants to use. If the server does not support the RFC-2228 extensions, it responds with reply code 500. If the server can accept the security mechanism, but requires further security data (e.g. a user name or password), it will respond with reply code 334. If the server can accept the security mechanism without requiring additional security data, it will issue reply code 234. The RFC also defines several reply codes for indicating the server is RFC-2228 compliant but does not understand, or cannot respond to, the specified security mechanism.

There are two important points to note regarding the above. First, the server's ability to indicate it does not support the RFC-2228 extensions makes possible a client that supports a secure connection, but is backwards compatible with older FTP servers. Second, reply codes indicating a particular security mechanism is not supported allow the client and server to continue negotiating until a mutually acceptable security mechanism is found.

Note that RFC-2228 does not specify any particular security mechanism. Instead, it specifies the use of the ADAT command to provide authentication and security data. Security data can be any arbitrary data block, meaning a server can accept a password, a certificate or key, a token identifier or any other security data.

Finally, RFC-2228 allows the client and server to negotiate the characteristics of both the control and data channels. A channel can be set up in one of four modes: unprotected, integrity protected, confidentiality protected and privacy protected. Unprotected is a normal clear-text data stream. Integrity protection ensures the data stream cannot be tampered with (e.g. packets or commands inserted), while confidentiality protection encrypts the data stream without providing any integrity checking. A privacy protected data stream uses both encryption and integrity checking. The control and data channel characteristics are negotiated separately. For example, the control channel may be privacy protected while the data channel is only integrity protected.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

OK, there's a problem… What should I do?

Several techniques can be used to protect against network eavesdropping. Each approach has different strengths, so before examining some of the available alternatives, it is useful to understand exactly what we want to accomplish. One approach for doing so is to determine "musts and wants" - what a solution must do in order to solve our problem and what we might want, but not require, the solution to do.

The "musts" are simple. An appropriate solution must:

- **Avoid exposing the username and password.** This may be done a number of ways, such as encrypting the username and password within the control channel, encrypting the entire control channel or employing an alternate method of authentication, such as a digital certificate. Choosing the best alternative will depend on circumstance, however any system that does exposes the username and password does not meet our needs.

The "wants" are more complex and will depend on the specific requirements of a site. "Wants" might include:

- **RFC-2228 Compliance.** Many organizations have started to use "open" software – software that uses publicly defined protocols and interfaces as opposed to proprietary implementations. RFC-2228 represents an open standard for securing FTP.

- **Encryption of the data connection.** For sites serving sensitive data, it may also be desirable to protect the data connection as well as the control connection. For some sites, it may be sufficient to manually encrypt the data on the server before it is transferred, decrypting it once it's been received on the client machine. For others, this process may need to be automatic.

- **User Transparency.** There are two reasons why an organization may want a solution that introduces no changes from the end user's perspective. One is the cost associated with retraining staff. Another is that a solution that is perceived as overly complex or cumbersome, particularly by non-technical users, might not be used.

- **Interoperability with existing FTP clients.** Many organizations have developed sophisticated and complex infrastructures around FTP and are not willing to incur the costs of redeveloping these interfaces. In some cases, these interfaces may be part of a commercial product that an organization may not have the ability to modify. Interoperability may also be a requirement when an organization cannot control the client software, as is the case with a public FTP site or when an appropriately enabled client is not available.

Additional requirements might include the availability of technical support, support for specific operating systems, the use of specific encryption or authentication mechanisms, etc.

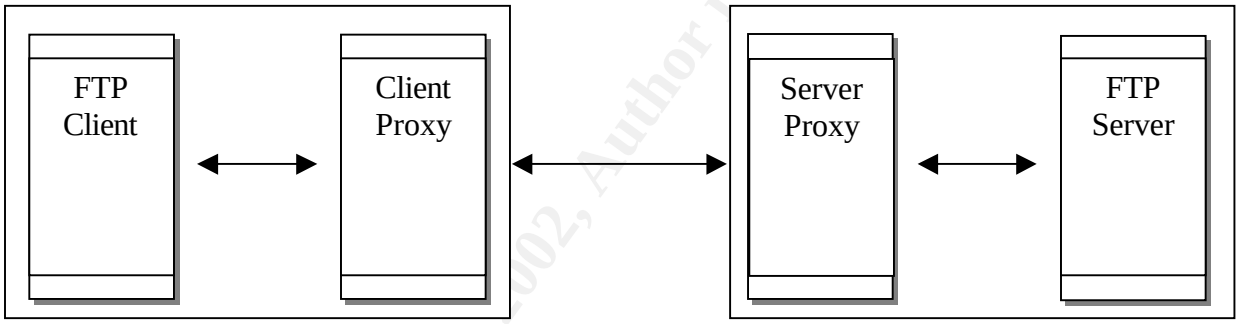
© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Approaches to Implementation

In practice, there are two general ways secure FTP services have been implemented. The first is to simply replace the standard FTP client and server software with versions that are encryption aware and, ideally, RFC-2228 compliant. This approach provides the simplest deployment, but potentially suffers from several drawbacks.

- A new software product may contain exploitable vulnerabilities.
- The new software may not have all of the features and flexibility of the product it's replacing.
- The new product may require specific client software to be used, potentially requiring retraining or the redevelopment of client side scripts.

The second approach is to use a proxy service. In this case, an additional piece of software is loaded on the client and server machines, through which the standard FTP processes communicate. This approach is illustrated the following diagram.



The advantage to this approach is that the client and server software can usually remain as is, with the implementation potentially transparent to the end user. Further, since the proxy software is actually listening on the FTP port (port 21), it is often possible to configure the FTP server to listen on some other, high-numbered port. This is advantageous in that it allows the server to be run as an unprivileged user.

There are some drawbacks, however:

- The proxy software may contain exploitable vulnerabilities.
- The initial software configuration is more complex, particularly for end users.
- Encrypting the data channel is often difficult if the proxy software is not specifically FTP aware.
- This approach may impact performance as two layers of software are now involved.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Example Implementations

The products described below are representative of the two different approaches to solving this problem. This list of products is by no means exhaustive. Inclusion in this discussion does not necessarily constitute a recommendation for a specific product, nor does exclusion imply any defect. The products below have been chosen because they are in common use and illustrate a number of the pros and cons of each approach.

SFTP (SSH)

The Secure Shell (SSH) protocol and suite of products are becoming increasingly popular, primarily as a replacement for Telnet and rsh. (These programs suffer from the same clear-text authentication issues as FTP.) SSH is now included by default with a number of operating systems and is readily available for many others.

SSH provides three ways in which can be used to replace or supplement FTP.

- **Scp** – scp is a replacement for the Unix remote copy command ("rcp"). The syntax and user interface of scp are very different from FTP and, as a result, it cannot be used as a simple replacement for a standard FTP client. (Scripts may need to be re-coded, users re-trained, etc.)

- **SSH Tunnel** – SSH can be used as a proxy service for FTP. To do this, an SSH connection is created between the client and the server. The FTP client connects to the SSH process on the local machine, which encrypts the data and sends it to the server. The server is also running an SSH process which receives the data, decrypts it and passes it to the FTP daemon.

  The difficulty with this approach is tunneling the FTP data connection. Since the data connection uses a randomly selected high-numbered port, it is difficult to construct an SSH tunnel specifically for that connection. Thus, most FTP implementations based on SSH tunnels secure only the control connection.

- **Sftp** – The SSH suite of applications includes an FTP-like program called "sftp". Sftp looks and feels very similar to FTP, however there are some key differences. Sftp actually operates against the SSH daemon (sshd) on the server. When launched, sftp attempts to connect and authenticate against port 22, not port 21. The SSH daemon then launches the sftd-server process to handle the request. Once authenticated, data is then transferred across the established connection.

  These differences can introduce difficulties. First, sftp is not backward compatible with standard FTP servers, nor can a standard FTP client connect to the SSH daemon. Second, anything that uses FTP programmatically may fail, since the same command set is not implemented. Finally, firewalls may need to be reconfigured to allow port 22 transactions. On the plus side, sftp is similar enough to FTP that retraining costs should be minimal and its widespread adoption makes support easier.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

SSH2, the current version of the SSH protocol is described in four IETF drafts: "SSH Protocol Architecture" [13], "SSH Transport Layer Protocol" [14], "SSH Connection Protocol" [15] and "SSH Authentication Protocol" [16]. Sftp is described fully in the Internet Draft "SSH File Transfer Protocol" [17]. A free implementation of the SSH protocol and application suite can be found at http://www.openssh.org.

SafeTP

SafeTP is a proxy-based mechanism for securing standard FTP applications. On the client, SafeTP uses the "Windows Socket 2 Service Provider Interface" (Winsock SPI) to intercept network requests from the FTP client. It then encrypts these requests and relays them to the server. On the server side, a new process, sftpd, is installed. The server is configured to launch sftpd in place of the normal FTP daemon. Sftpd then listens on port 21, while the original FTP daemon is configured to listen on some other port.

The SafeTP client proxy uses the RFC-2228 security extensions to determine if the server is running a SafeTP server. If not, SafeTP runs as transparent proxy, passing the FTP sessions unaltered. Similarly, when the SafeTP server receives a connection, it responds in the standard fashion. If the initial request from the client is a "USER" or "ACCT" command, the proxy assumes the client is insecure and proceeds to act transparently. If the initial request is an "AUTH" command, the SafeTP server proceeds with the RFC-2228 negotiation.

SafeTP has several advantages over other approaches:

- It provides transparent security for any FTP client under Microsoft operating systems.
- Installation is done through a standard Install Shield package, simplifying installation for end users.
- Proxy functionality allows the continued use of the existing FTP server, including all configurations (e.g. a chroot jail, if one has been implemented).
- SafeTP can be configured to secure both the control and data connections.
- SafeTP uses fairly sophisticated key exchange and data exchange mechanisms to guard against sever impersonation, replay attacks, etc.

There are some drawbacks as well. In particular, there is no transparent client implementation for operating systems other than Windows. Perhaps more problematic, client and server private keys are stored unencrypted. This is done for ease of management, however it means that organizations implementing SafeTP rely heavily on the physical security of their machines.

The complete specification of SafeTP can be found in the document "SafeTP: Transparently Securing FTP Network Services". [18] SafeTP can be obtained from http://www.cs.berkeley.edu/~smcpeak/SafeTP.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

SRP FTP

The Secure Remote Password (SRP) protocol is an attempt to address weaknesses in previous authentication schemes without sacrificing performance. The SRP project has developed a number of applications based around the protocol, including an implementation of Telnet and FTP. The use of SRP is becoming more widespread and a wide variety of SRP-enabled applications are becoming available.

As opposed to the previous two examples, SRP FTP is a drop-in replacement for an existing FTP deployment. To enable encryption, an SRP-enabled client and server are required. The SRP FTP server is RFC-2228 compliant, however, meaning that non-SRP clients are able to connect to it using the standard clear-text connections.

SRP implements strong password authentication. In many products (SSH being one example), although the control channel is encrypted, the password being sent is often weak. Brute force attacks against the server's password repository or a captured data stream can potentially reveal the password. SRP is designed to guard against this by exchanging a one-way key built from the user name and password. The key is calculated in such a way that, even if it somehow should be obtained from the network session, it is still not feasible to derive the user name and password.

Although more SRP-enabled applications are coming available, one concern for system administrators may be finding an implementation that supports both SRP and the desired feature set. This is likely to be more of an issue when evaluating FTP servers than with clients, where a system administrator may have a long list of security and management features they are looking for.

The SRP protocol is fully described in RFC-2945, "The SRP Authentication and Key Exchange System". [19] SRP FTP can be obtained from http://www-cs-students.stanford.edu/~tjw/srp.

TLS FTP

The Transport Layer Security protocol (TLS) is based on version 3 of Netscape's Secure Socket Layer (SSL) protocol. TLS is not compatible with SSL v3.0, however it does provide a mechanism by which a TLS implementation can support an SSL connection. Like SRP, TLS is a protocol around which specific applications are built. The Internet Draft "Securing FTP with TLS" [3] describes how FTP can make use of TLS.

Like SRP FTP, TLS-enabled FTP products are drop-in replacements for existing FTP implementations. TLS FTP is RFC-2228 compliant and will support non-TLS enabled clients. Although similar in approach to SRP FTP, there are several features that distinguish TLS-based implementations:

- TLS supports the use of X.509 certificates and a public key infrastructure, providing a scalable and industry standard approach to key management.

- Since it is based on, and backward compatible with, Netscape's SSL implementation, a TLS-enabled FTP server can be transparently supported through web browsers.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

- The widespread use of HTTPS makes it likely that TLS will be more easily and widely accepted than less well-known protocols.

Perhaps the biggest shortcoming of TLS FTP is the lack of available implementations. This can be expected to improve over time, particularly if the current Internet Draft document is adopted as a standard.

The TLS protocol is fully described in RFC-2246, "The TLS Protocol". [20] FTP under TLS is defined in the Internet Draft "Securing FTP with TLS". [3]

References

[1] Postel, J., Reynolds, J. "File Transfer Protocol", RFC 959. October 1985.
http://www.ietf.org/rfc/rfc0959.txt?number=959

[2] Bellovin, S. "Firewall Friendly FTP", RFC 1579. February 1994.
http://www.ietf.org/rfc/rfc1579.txt?number=1579

[3] Ford-Hutchinson, Paul, Carpenter, Martin, Hudson, Tim, Murray, Eric, Wiegand Volker, "Securing FTP with TLS" (Work in progress) October 3, 2001.
http://search.ietf.org/internet-drafts/draft-murray-auth-ftp-ssl-08.txt

[4] CERT Coordination Center. "Multiple Vulnerabilities in WU-FTPD", CA-2001-33. November 29, 2001. http://www.cert.org/advisories/CA-2001-33.html

[5] Adams, Joseph. "FTP Server Security Strategy for the DMZ". June 5, 2001.
http://rr.sans.org/securitybasics/DMZ.php

[6] Brennan, Michael. "Guest HOWTO". September 15, 1995.
http://www.wu-ftpd.org/HOWTO/guest.HOWTO (July 28, 2000.)

[7] CERT Coordination Center. "Anonymous FTP Guidelines". 1995.
http://www.cert.org/tech_tips/anonymous_ftp_config.html (May 4, 2001.)

[8] *Hobbit*. BuqTraq. July 11, 1995
http://www.securityfocus.com/archive/1/3488

[9] Song, Dug. "dsniff"
http://www.monkey.org/~dugsong/dsniff/

[10] van Hauser. "THC-Parasite"
http://www.thehackerschoice.com/releases.php?q=parasite

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

[11] Sipes, Stephen. "Why your switched network isn't secure". September 10, 2000.
http://www.sans.org/newlook/resources/IDFAQ/switched_network.htm

[12] Horowitz M., Lunt, S., "FTP Security Extensions", RFC-2228. October, 1997.
http://www.ietf.org/rfc/rfc2228.txt?number=2228

[13] Ylonen, T., Kivinen T., Saarinen, M., Rinne, T., Lehtinen, S., "SSH Protocol Architecture", Internet-Draft,draft-ietf-secsh-architecture-12.txt. (Work in progress) January 31, 2002.
http://www.ietf.org/internet-drafts/draft-ietf-secsh-architecture-12.txt

[14] Ylonen, T., Kivinen T., Saarinen, M., Rinne, T., Lehtinen, S., "SSH Transport Layer Protocol", Internet-Draft, draft-ietf-secsh-transport-12.txt. (Work in progress). January 31, 2002. http://www.ietf.org/internet-drafts/draft-ietf-secsh-transport-12.txt

[15] Ylonen, T., Kivinen T., Saarinen, M., Rinne, T., Lehtinen, S., "SSH Connection Protocol", Internet-Draft, draft-ietf-secsh-connect-15.txt. (Work in progress) January 31, 2002.
http://www.ietf.org/internet-drafts/draft-ietf-secsh-connect-15.txt

[16] Ylonen, T., Kivinen T., Saarinen, M., Rinne, T., Lehtinen, S., "SSH Authentication Protocol", Internet-Draft, draft-ietf-secsh-userauth-14.txt. (Work in progress) January 31, 2002.
http://www.ietf.org/internet-drafts/draft-ietf-secsh-userauth-14.txt

[17] Ylonen, T., Lehtinen, S., "SSH File Transfer Protocol", Internet-Draft, draft-ietf-secsh-filefxer-02.txt. (Work in progress) October, 2001.
http://www.ietf.org/internet-drafts/draft-ietf-secsh-filexfer-02.txt

[18] Bonachea, Dan., McPeak, Scott, "SafeTP: Transparently Securing FTP Network Services", U.C. Berkely Tech Report UCB/CSD-01-115. February, 2001.
http://sunsite.berkeley.edu/Dienst/UI/2.0/ncstrl.ucb/CSD-01-1152

[19] Wu, Tom. "The Stanford SRP Authentication Project". Updated August 3, 2001.
http://www-cs-students.stanford.edu/~tjw/srp/

[20] Dierks, T., Allen, C., "The TLS Protocol", RFC-2246. January, 1999.
http://www.ietf.org/rfc/rfc2246.txt?number=2246

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Last Updated: September 3rd, 2015

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Crystal City 2015	Crystal City, VAUS	Sep 08, 2015 - Sep 13, 2015	Live Event
SANS Network Security 2015	Las Vegas, NVUS	Sep 12, 2015 - Sep 21, 2015	Live Event
SANS Seoul 2015	Seoul, KR	Sep 14, 2015 - Sep 19, 2015	Live Event
Data Breach Investigation Summit & Training	Dallas, TXUS	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Baltimore 2015	Baltimore, MDUS	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Perth 2015	Perth, AU	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Tallinn 2015	Tallinn, EE	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS ICS Amsterdam 2015	Amsterdam, NL	Sep 22, 2015 - Sep 28, 2015	Live Event
SANS Bangalore 2015	Bangalore, IN	Sep 28, 2015 - Oct 17, 2015	Live Event
SANS DFIR Prague 2015	Prague, CZ	Oct 05, 2015 - Oct 17, 2015	Live Event
SANS Seattle 2015	Seattle, WAUS	Oct 05, 2015 - Oct 10, 2015	Live Event
SANS Tysons Corner 2015	Tysons Corner, VAUS	Oct 12, 2015 - Oct 17, 2015	Live Event
SOS: SANS October Singapore 2015	Singapore, SG	Oct 12, 2015 - Oct 24, 2015	Live Event
SANS Gulf Region 2015	Dubai, AE	Oct 17, 2015 - Oct 29, 2015	Live Event
SANS Cyber Defense San Diego 2015	San Diego, CAUS	Oct 19, 2015 - Oct 24, 2015	Live Event
SANS Tokyo Autumn 2015	Tokyo, JP	Oct 19, 2015 - Oct 31, 2015	Live Event
SANS South Florida 2015	Fort Lauderdale, FLUS	Nov 09, 2015 - Nov 14, 2015	Live Event
SANS Sydney 2015	Sydney, AU	Nov 09, 2015 - Nov 21, 2015	Live Event
SANS London 2015	London, GB	Nov 14, 2015 - Nov 23, 2015	Live Event
Pen Test Hackfest Summit & Training	Alexandria, VAUS	Nov 16, 2015 - Nov 23, 2015	Live Event
SANS Hyderabad 2015	Hyderabad, IN	Nov 24, 2015 - Dec 04, 2015	Live Event
SANS Cape Town 2015	Cape Town, ZA	Nov 30, 2015 - Dec 05, 2015	Live Event
SANS San Francisco 2015	San Francisco, CAUS	Nov 30, 2015 - Dec 05, 2015	Live Event
SANS Milan 2015	OnlineIT	Sep 07, 2015 - Sep 12, 2015	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced