GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials: Network, Endpoint, and Cloud (Security 401)"
at http://www.giac.org/registration/gsec

Joe Manek
15 December 2003
GSEC Practical Assignment, ver 1.4b Option 1

Securing FTP. Then and now.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Abstract

File Transfer Protocol (FTP) is one of the most highly utilized protocols in computing today. It's a protocol that neither the Internet nor most corporate intranets could long do without. However, for all FTP's use and ubiquity it remains one of the most under secured and over exposed capabilities being used today. This paper will provide some historical perspective of where FTP came from and where it stands today, as well as to identify some methodologies for controlling and securing FTP in a Unix environment, from a systems administrator's perspective. Additionally, several feature enhanced FTP replacements will be highlighted.

In the beginning

FTP has been around since very early in computing. FTP began as a need to easily and effectively distribute data and research information between continually increasing numbers, types and locations of computers making up the Advanced Research Projects Agency's Network (ARPANET). The Internet paper "A brief History of the Internet"[1] provides a nice perspective of the work done for the Defense Advanced Research Projects Agency (DARPA), which led to the ARPANET and the need for networking protocols.

Within the research community the ability to easily and reliably transfer files from host to host became a key requirement. As the networks grew, so did the difficulty and complexity of transferring data from one host to another. A File Transfer mechanism was needed to be able to abstract the underlying differences between different computer hosts and provide a simple user interface. The 1st Request For Comments (RFC[2]) pertaining to this File Transfer Protocol was [RFC-114], introduced in 1971 as a File Transfer mechanism "between two hosts at M.I.T. the GE645/Multics and a PDP-10/DM/CG-ITS (and possibly Harvard's PDP-10)"[3].

Numerous RFCs have since been published pertaining to FTP and are continuing today. . A complete and searchable listing of RFCs can be accessed at http://www.faqs.org/rfcs.

Since those early dark-days the File Transfer Protocol has come a long ways. The current standard Internet standard for FTP is described by [RFC-959] with additional features defined in subsequent RFCs. Such as:

- [RFC-1123], a broad ranging document containing many FTP related recommendation, clarifications and improvements to [RFC-959]. Of note amongst the many enhancements is the addition of the PASV command, which

[1] http://www.isoc.org/internet/history/brief.shtml
[2] http://livinginternet.com/i/ia_rfc_invent.htm
[3] http://rfc-114.rfc-list.org/rfc-114.htm (page 1)

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

allows the client to determine the data port number opened by the server for data transfer.

- [RFC-1579], defines enhancements that allow FTP to work more easily through firewalls. This RFC suggests a change for an implementation of FTP that would replace the PORT command with the PASV command, thereby making the data connection an outbound connection and much easily to implement and control via packet filters and firewalls.
- [RFC-2228], defines security related enhancements addressing both confidentiality and integrity. These security extensions and enhancements provide the ability to negotiate a mutually agreed upon Authentication/Security Mechanism between the client and the server. This capability not only provides for the secure transfer of encrypted data but also the ability to authenticate the client to the server thereby assuring the client machine is who it claims to be.
- [RFC-2428], defines extensions for IPv6 and NAT. These extensions provide several 'extended' FTP subcommands that enable and support use on both IPv4 and IPv6 environments.
- [RFC-2577], identifies further security considerations when implementing FTP, such as protecting against a "bounce attack". A "bounce attack" makes use of the ability to designate an IP address and port number using the "PORT" command of FTP. This ability allows FTP to be used to direct an attack to a legitimate service on a target machine, such as SMTP or NNTP. A 'forged' file containing valid input to the service being attacked could be sent to the service in this way. [RFC-2755] specifies that the 'data' connection of FTP not be allowed to specify or open any port numbers below 1023 (ports below 1023 are considered well-known ports). Note: this does not protect well-known ports used above 1023.
- [RFC-2773], describes an experimental Encryption implementation using KEA and SKIPJACK employing the security extensions defined by [RFC-2228].

To secure or not to secure

FTP evolved and matured in the era before the Internet we know today came to be. There wasn't a vast worldwide network of interconnected systems, a large number of which whose owner's sole purpose it seems is to break into your system and look at, if not steal or destroy your data. The security characteristics of the early FTP relied mostly upon the inherent security of the ARPANET itself, and those who used it.

Little attention to security was given in any of the early RFCs pertaining to FTP, past the admonishment that users protect their passwords from exposure, as indicated by this excerpt about the PASSword argument from [RFC-959],

```
The argument field is a Telnet string specifying the user's
password.  This command must be immediately preceded by the
user name command, and, for some sites, completes the
user's identification for access control.  Since password
information is quite sensitive, it is desirable in general
```


© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

> to "mask" it or suppress typeout. It appears that the server has no foolproof way to achieve this. It is therefore the responsibility of the user-FTP process to hide the sensitive password information.[4]

FTP, at [RFC-959] provides no built-in ability for securing or protecting your data. Here are some of the weaknesses:

- User authentication consists of supplying a user account and password, both of which transit the network in clear-text, easily obtained with proper tools and or access points.
- No authentication of the client other than with the password, thereby assuming the possessor of this information is authorized
- No limit on the number of attempts to connect which lends itself to password quessing
- Difficult to secure with perimeter defenses such a firewall. Due to the dual-channel nature of FTP it is difficult to secure the in-bound data channel since the port number is determined at each transfer. Conceptually one would have to allow all in-bound high-port numbers in order to provide FTP services.
- Limited logging. Logging typically consists of supplying an operand, such as '-d' or '-l' at startup of the 'ftpd' daemon, which logs each attempt noting the IP address of the client and possibly the operation (put, get, etc.)

Common techniques for making FTP more secure.

Over time, given FTP's inherent insecurity system administrators have come up with ways to strength security aspects and diminish it's use as a tool for nefarious purposes. Below are just a few of these techniques.

Anonymous FTP

Most Unix implementations of FTP include 'anonymous' FTP, and although slightly different from Unix to Unix, a 'man' on ftpd is usually all that's needed for setup.

Anonymous FTP typically has the user authenticate using a shared account name of 'ftp' or 'anonymous' and a non-authenticated password of any valid string, typically the user's email address. Some implementations attempt minor validation of this string by looking for a "@" or something resembling an account name. At first glance, this might seem to be less secure than FTP requiring both a unique account and a valid password combination. However, a unique feature of anonymous FTP is the concept of an FTP jail or fenced sandbox. This jail is implemented with the 'chroot' capability of most Unix's. 'chroot' basically defines a directory structure relative to a defined 'root', which is determined by the system admin setting up the anonymous FTP server. This effectively boxes the FTP user in and eliminates the ability to use FTP to traverse up the directory

[4] http://rfc-959.rfc-list.org/rfc-959.htm (page 26)

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

structure past this relative 'root', thus preventing the 'curious' user from playing outside of their sandbox.

Although gaining some level of protection for the rest of the system, anonymous FTP also has drawbacks. There is no granularity of control. Everyone is user 'ftp' or 'anonymous'. Everyone uses the same directory structure defined by the 'chroot' command. Users are typically instructed ahead of time to navigate to a specific subdirectory or set of subdirectories before placing or retrieving data. These directory structures and the files contained therein can be hidden from the FTP user's view using the following tasks:

- Set the permissions for the directory being used to receive files to '130', Octal. An 'ls –l' on the directory name should produce permission of, 'd--x--wx---'
- Create a new group and make it the default group for the 'ftp' account. The 'ftp' account should be the only account defined to this group.
- Make this group the group-owner of the directory.

When the user authenticates as account 'ftp', the user will be placed in the HOME directory designated for the 'ftp' account. Remember, this directory needs to be read-only to the 'ftp' user and associated group so that subdirectories can't be removed or renamed. The user still will be able to 'put' and 'get' files but will not be able to list them via the 'ls' or 'dir' commands. A "550 no files found" message will be returned in response to either of these commands. The 'cd' command will still allow the user to navigate to any subdirectories to which the 'ftp' account or it's group has the proper access. The name and location of files and directories must be known before they can be retrieved. This doesn't prevent someone from guessing for file names, but it does make it appreciably more difficult than merely listing them.

Limiting access to FTP only

A restriction of FTP is that the account used must meet the following criteria.

1. The username must be in the password database `/etc/passwd` and must *not* have a null password. The client must provide a password before any file operations may be performed.
2. The username must *not* appear in the file `/etc/ftpusers`, which contains a list of users who aren't allowed to use of `ftp`. One username is listed per line. If this file is missing, *anyone* on the local system may access `ftp`.
3. The user must have a standard shell (i.e. one listed in `/etc/shells`).
4. If the username is `anonymous` or `ftp`, an anonymous `ftp` account must be present in the password file (user `ftp`). In this case, the user can log in by giving any password (by convention, users give the name of the client host).

With out further controls, a user can use the same credentials to authenticate and gain command-line access even though they are 'supposed' to be FTP-only. The following steps can be taken for restricting access.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

- Create a separate home directory for each FTP-only user, designated in their '/etc/passwd' entry.
- Limit the user's authority as tightly as possible only to those directories to which they should have FTP access.
- Create a 1-line shell script as follows: "#!/bin/ksh". with 755 permissions, owned by 'root' and placed in a 'root' owned directory, such as '/usr/local/bin'.
- Add an entry for the full pathname of the 1-line shell script to /etc/shells. This makes it a valid script that can be used as 'shell', a requirement for FTP. It is not a valid shell for command-line access and as such will merely exit. You can, if you wish add some logic to put out a nasty-gram to the user, letting them know this account is for FTP only. Although, silence is usually golden, and typically more secure.
- Make this 1-line shell the default shell for the user.

Be aware that this technique does nothing to prevent the FTP user from using built-in FTP command such as 'cd' to traverse any directory tree to which they have 'execute' access, retrieve any files to which they have 'read' access and remove or rename files to which they have 'write' access. This technique merely allows you to create FTP accounts that can't be used to un-intentionally authenticate command-line access as well.

TCP Wrappers

TCP Wrappers can be used to help authenticate clients to the FTP server by providing a level of assurance that a client machine's IP address is not being 'spoofed'. In order to accomplish this, Wrappers should be run in 'PARANOID' mode. 'PARANOID' mode takes all in-coming connections and does a 'reverse DNS lookup' on all inbound IP connections. It then takes the 'hostname' returned by the lookup and does a 'forward DNS lookup' and compares the two IP addresses. If they aren't the same the request is rejected. Even when not ran in PARANOID mode TCP Wrappers can provide additional levels of logging.

Adding TCP Wrappers to FTP is as simple as;

1. Installing TCP Wrappers. Not necessarily trivial but should be doable on almost all variants of Unix, new and old.
2. Replace the /etc/inetd.conf entry for FTP with the Wrapper version.
3. Add an 'ftpd.in' entry in /etc/hosts.allow to allow specific or ranges of IP addresses to be allowed to FTP.
4. Place a "PARANIOD" entry in the /etc/hosts.deny, either specifically for 'ftpd' service being ran, or with "ALL" for all services.

Be warned that running Wrappers in "PARANOID" mode can deny access to legitimate users if DNS is not properly configured or working properly. Additionally, DNS itself is vulnerable to several methods of attack and as such could be compromised or disrupted.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

DNS vulnerabilities and their solutions can be easily located at http://www.cert.org/advisories. An excellent overview of TCP Wrapper function, installation and configuration can be found in the paper "Wrap a Security Blanket Around your Computer"[5] written by Lee E. Brotzman.

FTP replacements

Many FTP replacements are available via both Open Source and commercially. These typically offer similar feature and usability enhancements over [RFC-959]. Otherwise why bother. Right? I'll highlight the features of a couple of leaders.

WU-FTPD

The clear leader of the Open Source pack is wu-ftpd, available from ftp://ftp.wu-ftpd.org/pub/wu-ftpd/. wu-ftpd, also known as Wuarchive-ftpd, is replacement ftp daemon originally developed at the University of Washington by Chris Meyers and Bryan D. O'Connor. It is now supported by the "Wu-ftpd Development Group", which maintains a number of mailing lists, found at http://www.wu-ftpd.org/mailinglists.html.

wu-ftpd is written to the [RFC-959] standard as well as updated by [RFC-1579] and [RFC-2228]. For those interested in some light reading, a detailed listing and contents of wu-ftpd related RFCs is accessible at http://www.wu-ftpd.org/rfc/.

wu-ftpd supports 3 types of FTP.

1. anonymous FTP. Your basic anonymous FTP using a user name of 'ftp' or 'anonymous' against an account name of 'ftp'. The user is restricted the 'ftp' account's HOME directory via 'chroot'.
2. real FTP. The normal FTP everyone's familiar with where a real user authenticates with a real password.
3. quest FTP. Guest FTP operates the same as 'anonymous' FTP in that once a real user, with a real account, password and HOME directory, authenticate, they are placed in a 'chroot'ed jail of their HOME directory. Thereby eliminating the exposure to the rest of the system to any curious cruising around. ftp://ftp.fni.com/pub/wu-ftpd/guest-howto/ provides an excellent tutorial for setting up a guest environment. Be prepared for a little extra work when creating accounts, because guest FTP accounts require the same amount of Operating System specific setup for each real user account that anonymous FTP does for the 'ftp' user account.

'ftpaccess', located in /etc can be used to configure and control wu-ftpd's behavior. A myriad of parameters and options allow the administrator to tightly control most aspects of the FTP server. An explanation of 'ftpaccess' and wu-ftpd's configuration is far

[5] http://www.linuxjournal.com/article.php?sid=2180

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

beyond the scope of this paper but can be examined further at http://www.wu-ftpd.org/man/ftpaccess.html.

ProFTPD

ProFTPD, a free FTP server (GPL), available from http://www.proftpd.org/ is also highly configurable feature-rich offering. Developed from the ground-up as a very modular, easily extensible alternative to wu-ftpd, ProFTPD offers the same features and capabilities as wu-ftpd with some notable improvements. Such as:

- Per directory ".ftpaccess" configuration similar to Apache's ".htaccess".
- Anonymous FTP root directories do not require any specific directory structure, system binaries or other system files.
- Runs as a configurable non-privileged user in stand-alone mode in order to decrease chances of attacks which might exploit its "root" abilities. Note: This feature is dependent on the capabilities of the host Unix system.

These features and others are highlighted at http://www.proftpd.org/features.html.

PureFTPd

PureFTPd, a free (BSD) FTP server, available from http://ww.pureftpd.org/, boasts of a simple 'no frills' implementation with a strong emphasis on security, flexibility and extensibility. Some of PurFTPd's features are listed below, the remainder can be found at http://www.pureftpd.org/index.shtml.

- All accounts can 'chroot'ed by default
- Supports LDAP authentication and a large number of crypto hashing algorithms, such as; Plaintext, Crypt, MD5, SMD5, SHA and SSHA.
- Ability to throttle bandwidth for both downloads and uploads.
- Provides a utility, 'pure-ftpwho', that provides real-time reports of who's doing what on the FTP server, including bandwidth usage. Can be tailored to produce html, XML or text reports.
- One of the most complete implementation of the FTP protocol specification, plus modern extensions.

Although not as widely used (yet) as the other FTP servers described in this paper, it has some impressive feature and warrants a close look when deciding on an Open Source FTP server.

NcFTPd

NcFTPd, a commercial FTP server, available from http://ww.ncftp.com/, has many of the same features as the non-commercial offerings listed above. It also has one that the others don't. Support. Albeit only e-mail and fax support, it's still more than is typically provided from the Open Source community. Some additional features are:

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

- Easy installation and setup. Provides a default installation that drops-in with out much effort.
- Built-in 'compress' or 'gzip' compression and the ability to retrieve directories as entities, using 'tar'.
- Utilities to easily monitor and log activities on the server, in real-time.
- Ability to filter and control file and pathnames, which can be used to prevent un-printable or '*.dotfiles'* from being created.
- Boasts of several performance enhancements that increase throughput and decrease resource consumption.
- Comes with several optimized client tools

A more complete list of features for NcFTPd can be found at http://www.ncftp.com/ncftpd/features.html.

SUMMARY

Everyone in computing, whether on the Internet, a corporate Intranet or as a home user, uses the File Transfer Protocol (FTP). It began as a need for a platform independent mechanism for transferring data between research hosts at M.I.T. It's been around since before Al Gore invented the Internet. It's also been the object of much updating and enhancement, in the form of many Requests For Comments (RFC). The RFC describing the current FTP standard is [RFC-959], although most modern implementations of FTP have also incorporated more recent RFCs, focusing somewhat on usability, but more so on security enhancements and extensions.

In today's target rich environment, with bad guys lurking around every corner, it's important to harden your FTP environment. This paper briefly describes and provides some references and URLs to several possible ways of making your FTP server a little less of a target. Such as; Implementing "anonymous" FTP where appropriate, restricting or eliminating cross-protocol use of FTP account authentication and using TCP Wrappers as a tool to further secure, control and log your FTP server's activities.

Using an FTP replacement was discussed as a way to provide greatly enhanced control, monitoring and security characteristics over the standard [RFC-959] implementation. There are many FTP server replacements available today. A short search on Google will reveal many to choose from. 3 Open Source and 1 commercial FTP servers were highlighted as a sampling.

Lastly, do not assume your FTP servers are secure. If you haven't, take some time and perform some basic research and determine what your FTP environment looks like. Check to see if simple changes will make it more secure. Or, if it's appropriate, replace your aging FTP server software with one of the more modern, feature rich and secure variants available.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

References

[RFC-114] A. Bhushan, "A File Transfer Protocol"
RFC 114, April 1971.
http://rfc-114.rfc-list.org/rfc-114.htm

[RFC-959] J. Postel, "File Transfer Protocol"
RFC 959, October 1985.
http://rfc-959.rfc-list.org/rfc-959.htm

[RFC-1579] S. Bellovin, "Firewall-Friendly FTP"
RFC 1579, February 1994.
http://rfc-1579.rfc-list.org/rfc-1579.htm

[RFC-2228] M. Horowitz, S. Lunt, "FTP Security Extensions"
RFC 2228, October 1997.
http://rfc-2228.rfc-list.org/rfc-2228.htm

[RFC-2577] M Allman, S Ostermann, "FTP Security Considerations"
RFC 2577, May 1999.
http://rfc-2257.rfc-list.org/rfc-2257.htm

[RFC-2773] P. Housley, P. Yee, W. Nace, "Encryption using KEA and SKIPJACK"
RFC 2773, February 2000.
http://rfc-2773.rfc-list.org/rfc-2773.htm

Liener, Barry M., Cerf, Vinton G., Clark, David D. Kahn, Robert E., Klienrock Leonard, Lynch, Daniel C., Postel Jon, Roberts, Larry G., Wolf, Stephen "A Brief History of the Internet"
http://www.isoc.org/internet/history/brief.shtml

Chapter 22, "Wrappers and Proxies" of O'REILLY's "Practical UNIX & Internet Security" contains a nice summary of Wrapper capabilities. Available online at
http://www.busan.edu/~nic/networking/puis/ch22_03.htm

Chapter 32, "Section VI – Unix Security: Step-by-Step" of The SANS Institutes' "SANS Security Essentials with CISSP CBK Version 2.1"

Brotzman, Lee E., "Wrap a Security Blanket Around Your Computer"
Friday, August 01, 1997. Published in Issue 40 of LINUX® Journal
http://www.linuxjournal.com/article.php?sid=2180

©SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.