Global Information Assurance Certification Paper

Enterprise Considerations to Updating Application Use of FTP

GIAC Security Essentials
Certification (GSEC)
Practical Assignment
Version 1.4b

Option 1 - Research on Topics
in Information Security

Submitted by: Robert M. Ruhge
Location: Orlando, FL
Date Submitted: September 20, 2004

Paper Abstract:
Often, enterprise applications exchange files with each other using ftp which has serious vulnerabilities. There are several approaches to increasing the security level of transferring files between systems. Modifying system settings, obtaining a secure ftp application, or even replacing ftp with another file exchange application. Some logical approaches may actually add vulnerabilities and management complexity – inadvertently increasing risk and cost. Additionally, there are many factors to consider when updating an enterprise with improvements to ftp security. Compatibility, encryption level, level of effort and scope all need to be addressed before an enterprise-wide update may be implemented.

Table of Contents

Abstract/Summary

Often, enterprise applications exchange files with each other using ftp which has serious vulnerabilities. There are several approaches to increasing the security level of transferring files between systems. Modifying system settings, obtaining a secure ftp application, or even replacing ftp with another file exchange application. Some logical approaches may actually add vulnerabilities and management complexity – inadvertently increasing risk and cost. Additionally, there are many factors to consider when updating an enterprise with improvements to ftp security. Compatibility, encryption level, level of effort and scope all need to be addressed before an enterprise-wide update may be implemented.

1. Introduction

For decades, ftp has provided a simple and efficient solution for moving files from one computer to another. Large, enterprise applications systems sometimes use software that performs file transfers by making a call to ftp. In these enterprise applications, files may be ftp copied from a server to a user or from a server to another system for additional processing. In turn, after processing is complete, the file may again be ftp copied to another system for further processing or storage. Everything works well because ftp is standard across the enterprise.

What's the problem? Easy, ftp transmits usernames and passwords in clear text. FTP is listed in the SANS Top 20 Internet Security Vulnerabilities for this reason (SANS, 2004). Depending on the value of the data and the potential for loss, a more secure method of transferring the data may be necessary. A method to ensure who the sender and the receiver are (authentication), that the data had not been changed during transit (integrity) and that the information cannot be read by those that it was not intended for (confidentiality).

This paper serves to outline:

1) An approach for a system administration department to use in deciding which technology path to take to improve the security of files being exchanged between applications.
2) Factors that should be taken into account when analyzing a large enterprise with heterogeneous hardware, operating systems and applications.

It is not the objective of this paper to provide detailed information of how to accomplish hardening techniques or upgrade procedures. There are plenty of papers, guidelines and books available at SANS, on the web and in bookstores. However, references will be made to these supporting documents. Nor is it the objective of this paper to list and compare the many packages available for increasing the security of exchanging files. This would be a major study which

would quickly be out of date and exceed the page count limit of this practical assignment.

There are three high level approaches that may be taken towards improving ftp confidentiality, integrity and authentication – each with varying degrees of success towards that goal. These approaches are listed here and discussed in the following section. System administrators may choose to:

- Harden ftp system settings
- Replace ftp with a more robust ftp package
- Replace ftp entirely with other secure alternatives

Defining the approaches is relatively easy. Deciding which approach to use is a more difficult task. The decision is dependent on several enterprise factors. These factors have been grouped into the following four areas:

- System and software compatibility
- Security concerns
- Level of effort required to make the change
- Defining the scope of work to be performed

The next section of this paper discusses the three approaches to securing ftp. The subsequent section expands on the enterprise factors that would drive the solution towards one approach over another.

2. Approaches to Securing FTP

Hardening, replacing, or implementing alternatives to ftp may be measured in terms of ease of implementation and the increase in security. Each of the techniques below have been rated as high, medium, or low for implementation ease and security improvement.

2.1. Harden System Settings

The first and easiest way to improve ftp security is to leave ftp in place and tighten up configuration settings. Joe Manek, in his paper entitled “Securing ftp. Then and now.” (Manek, 2003) provides an excellent discussion on techniques for making ftp more secure. Hardening techniques listed in his paper are paraphrased below:

- Use ftp with the ‘chroot’ capability limiting an anonymous user’s access.
- Set directory permissions to ‘--x--wx---‘
- Create a new, default group for only the ‘ftp’ account
- Make ftp group the group-owner of the directory
- Ftp usernames in the password database without a null password.
- Make sure the file /etc/ftpusers lists all prohibited ftp users.
- All users have a standard shell.
- The anonymous or ftp usernames are in the password file.
- Configure TCP Wrappers to perform a reverse look-up on the clients IP address to prevent spoofing.

Additionally, Steve Tobias' paper entitled "Case Study in Secure File Transfer: implementing Secure ftp with SSL in a Healthcare Organization" provides steps for hardening ftp on Windows 2003. The techniques describe ways to improve authentication and access control as well as logging. However, not much more may be added to plain old ftp to increase integrity and confidentiality.

Ease of implementation – High. There are a number of checklists that may be found to make this task a snap.

Security improvement – Low. Limiting access, improving authentication and logging do not meet the need for strong authentication, data integrity and confidentiality.

2.2. Replace ftp with a more Robust package

A second approach is to replace ftp with a more robust ftp. There are several commercial and freeware/shareware programs available. Joe's paper lists several products and their attributes such as WU-ftpD, ProFTPD, PureFTPd, and Neftpd (Manek, 2003).

OpenSSH should be added to the list of possible replacements for ftp. It is a package of applications that includes SSH commands for initializing a secure shell session on another computer. SSH is intended as a replacement for telnet, rlogin, rsh, and rcp. Another part of OpenSSH is SSH2 which provides a secure ftp application called sftp. Like ProFTPD and PureFTPd, sftp supports encryption. SSH1 and SSH2 provide strong authentication and secure connections across the LAN (Boran, 2004). SSH1 and SSH2 in turn, rely on SSL to provide this confidential, authenticated and integrity based communication between applications (Tobias, 2004).

More information on SSH may be found at:
http://www.openssh.com/
http://www.ssh.com/
http://www.ietf.org/html.charters/secsh-charter.html
http://www.onsight.com/faq/ssh/

The public key authentication features of SSH2 may be used to provide strong authentication. When using sftp instead of ftp, the login session including the password, is encrypted. Outsiders will not be able to view passwords on the network. Once a sftp session is initialized, the same commands as ftp may be used - with a few exceptions (University, 2004).

Although all this sounds great, there is one side effect that may be overlooked. With sftp comes, as mentioned before, SSH. Now SSH brings additional end-user features that were never available under ftp. These features may not be desired for systems that only need to exchange files. Now an end-user may log

into another system and have all the commands of the operating system available. The net result is that the risks for authentication, integrity and confidentiality are reduced, but the risk of access control has increased.

Another side effect is the increase in public and private key management. Although initially handled efficiently by sftp and SSH, any changes to system names will generate warning messages that end-users and administrators will need to deal with.

Ease of implementation – Medium. There are a number of commercial and shareware products with installation scripts to make this easy. Some planning is necessary before implementation.

Security improvement – High. Improving authentication, data integrity and confidentiality are attainable with most of the products. Selecting the right product will take time.

2.3. Replace with other Secure file transfer Alternatives

Another option may be to eliminate ftp entirely with another application for transferring files. Secure file transfer may be accomplished with either SSH over SSL. This technique performs the same authentication and encryption as sftp using industry encryption algorithms such as RSA key exchange and Triple DES. OpenSSH supports 3DES, Blowfish, AES and arcfour. OpenSSH replaces rlogin, telnet, rcp and ftp with secure components for the same functions. Instead of performing ftp, applications may instead use secure copy (scp), (Open, 2004).

Ease of implementation – Medium. There are a number of places SSH may be obtained. Some planning is necessary before implementation. Changes to the application software will be necessary.

Security improvement – High. As above, improving authentication, data integrity and confidentiality are attainable with SSH.

Although not a replacement for FTP, there is a an enhancement to standard ftp (RFC 959) which allows the same ftp protocol to be sent over SSL. It is sometimes called ftpS, ftp-SSL and ftp over SSL (Enterprise, 2004). FTPS has a number of advantages over SFTP. It does not add unwanted SSH services such as scp and ssh that provide users with remote access. It uses the same ports as ftp – it simply establishes an SSL connection over the ftp port(s). This means that no new ports need to be documented and opened on firewalls. Lastly, applications may use the same old ftp calls as they always have with no software changes. Given this approach, the ease of implementation goes to “High” as minimal software changes are required. The improvement in security is “High” as well given the benefit of eliminating the potential of end-user logins.

3. Enterprise Factors to Consider

The analysis of this issue falls into four categories – Compatibility, Security, Level of Effort and Scope. Each of these areas is detailed in the paragraphs below.

3.1. Compatibility

In a large enterprise there may be some systems that already support a secure file transfer capability. SSH may be implemented because another software application requires it. Other systems form other vendors may have plans to improve their file transfer security in future releases.

Unfortunately, sftp is not compatible with ftp. They are not similar at all. If a software application must exchange files with one system that supports ftp, and another system that supports sftp (using SSH2), the application must maintain a compatibility look-up table to determine which form of ftp to use.

For example, if the central database had SSH2 installed, it would be able to fulfill a request for a file from another system by looking up the ftp destination ip address in a text file and determining if it is listed as supporting sftp. If the requesting machine does support sftp, then the central database application would fulfill the request securely through sftp. If the machine does not support sftp, then the central database application would fulfill the request insecurely - through ftp. The concept of integrating sftp is not difficult. However, the process of identifying and maintaining the list of machines that do and do not support sftp is awkward and requires human intervention as systems migrate from ftp to sftp.

The operating system is a definite factor regarding implementing SSH in the segments. IRIX, Solaris 8, Solaris 9 and Win2k have different options for SSH. OpenSSH, supports the widest variety of operating systems. Solaris 9 includes an installation of SSH2. There are other versions of SSH available that support a limited number of operating systems. Unfortunately, the OpenSSH organization does not have an implementation available for Windows machines.

Another compatibility issue that arises is that there are several providers of SSH and secure FTP. Few vendors support Unix and Windows as well as the various versions of operating systems on each. This raises several questions. What if all systems are not using the same software? Will they be able to exchange files or not? This must be tested before any enterprise-wide change is implemented.

The following SSH clients are recommended by OpenSSH for Windows machines but should be tested before implementing: PuTTY, TTSSH (SSH1), Cygwin, MSSH, Secure iXplorer GPL, WinSCP, FileZilla (Open, 2004). If that is not enough, SSH1 is not compatible with SSH2. Although, one source states, “SSH2 can be compatible with SSH1, but is NOT compatible by default.”

(Yamamoto, 1998). A future study could be performed to identify the compatibility of various implementations of sftp.

3.2. Security

The proper level of encryption must also be determined by the enterprise. SSL provides a variety of encryption algorithms for SSH. A single standard must be identified to 1) ensure a sufficiently strong algorithm and 2) assure compatibility across the enterprise.

Strengthening confidentiality through sftp may actually weaken access control and inadvertently make it easier for users to see files for which they are not authorized to view. Here's how. Secure ftp applications rely on the ssh command and therefore, everyone who uses sftp will be provided remote login capability through the ssh command. This means that users may now log into remote machines and begin viewing the files sftp'ed by other users as well as execute any Unix command on the remote machine. This new capability may not be desirable and should be thoroughly investigated before SSH is implemented across the enterprise. Integrity and confidentiality go up, but access control goes down – unless configuration changes are made to prevent this which is described as "a rather tricky thing to do right." (Barrett, 2000).

Commercial authentication software may be purchased to enforce post-login properties for the user environment such as variables, working directory and shell. Symark licenses PowerPassword which manages login and password policies across heterogeneous UNIX environments. It also keeps a centralized, audit trail of login activity (Symark, 2004).

In fact, it is possible to set up an account on a Windows machine and through cygwin perform an ssh to another system with ssh installed. If the username and password match, you're logged in! Granted, you must know the user name and password – however, this combined with some scripting and lax management of the target machine and a brute force attack could be launched. Other vulnerabilities may also be present. This would be an excellent subject for a future paper.

Strengthening confidentiality through sftp unfortunately thwarts the ability for perimeter defense tools to analyze content. With ftp, the control and data channels are separate. This allows firewalls, IDSs and other malware detection tools to analyze content and prohibit questionable content. Within a trusted environment, encrypting the data channel is not a problem. However, there may be installations where content analysis is necessary.

A better approach would be to encrypt the control channel of ftp and leave the data channel to be transmitted in the clear. This may be accomplished through ftpS (described in section 2.3). If data confidentiality is not a concern, encrypt

the control channel and leave the data channel in the clear so data may be analyzed by perimeter defense systems (Enterprise, 2004). Enterprise Distributed Technologies sells development and run-time licenses for their Java-based edtFTPjSSL ftpS library.

An excellent list of current client, server and library implementations may be found at http://www.ford-hutchinson.com/~fh-1-pfh/ftps-ext.html. There are at least 28 server implementations and 40 client implementations (Ford-Hutchinson, 2004).

3.3. Level of Effort

The level of effort required for upgrading may be profound. After discussing the replacement of ftp with sftp or scp with software developers, one estimate was roughly a week or two for the software changes. However, the complexity extends beyond software development alone. SSH cannot be implemented on systems without proper testing. Testing must be made on many different platforms and can take a considerable amount of time. Also, all changes must be thoroughly documented. Training must be invoked at the enterprise level to show administrators how to update the ftp/sftp compatibility tables. Training may be necessary for some end-users who manage their own systems and use ftp services. As well as training for the administrators using SSH on their servers.

Depending on the encryption algorithm used, performance may be an issue - especially on slower machines. An analysis may be necessary to determine if some systems will need to be upgraded or replaced. Encryption hardware is available to increase performance on existing systems.

3.4. Scope

Lastly, where does the effort end? SSH requires SSL. SSL is also used to perform HTTPS. Will HTTPS be a new requirement in the near future? If so, why not tackle this problem right now in addition to securing ftp?

There may be legacy systems that are near end-of-life. This may be a good time to retire them.

4. Conclusion

In improving the file transfer security that applications use, three approaches may be used. The first approach provides limited ftp authentication and confidentiality improvements. It is accomplished through hardening of ftp related system settings. It is the easiest and least expensive approach to securing ftp.

The second approach is to replace ftp with a robust shareware/freeware or commercial ftp package that provides strong authentication, confidentiality and

integrity. This comes at a cost. Even the free solutions may require changes to the application software.

The third approach is to replace ftp entirely with another file transfer application such as SSH. This requires software changes, application documentation changes and thorough testing. The products are free, but the implementation costs are high.

There is hybrid approach which has good security and allows for ftp to be used as it always had. Make no changes to the application software which calls ftp and implement the ftp protocol and commands over SSL (ftpS). There is a lot of activity on this and the following web site should be monitored for developments: http://www.ford-hutchinson.com/~fh-1-pfh/ftps-ext.html

Before deciding which approach to use, four factors should be considered. First, determine the compatibility issues. Determine which operating systems and versions are in use, which secure file transfer package supports them. If it does not support all operating systems, make sure it is compatible with the other vendor that is selected.

Second, determine the security requirements. Decide on the level of encryption, if SSH is used, determine if secure copy and secure shell should be unavailable to uses. Consider if firewalls and malware detection programs need to scan the data as it is being passed from system to system.

Third, determine how difficult it will be to make the change keeping in mind that software changes are only a part of the overall effort. Software upgrades, especially to encryption algorithms, may require hardware or even network upgrades to handle the extra burden of encrypting / decrypting and increased network traffic from the encrypted data.

Fourth, define the scope of work to be performed. There may be upgrades to other hardware or software components that may be easily performed while increasing ftp security.

All of these issues must be carefully analyzed before all enterprise applications may be upgraded to a more secure method of exchanging files. The bottom line is that it is an enterprise concern and should not be treated as a piece-meal approach.

References

"Open SSH Features." 22 Mar. 2004. URL: http://www.openbsd.org/cgi-bin/man.cgi?query=sftp&sektion=1 (10 Sept. 2004).

Barrett & Silverman "SSH: The Secure Shell (The Definitive Guide) - FAQs" O'Reilly. 15 Oct. 2000. URL: http://www.snailbook.com/faq/restricted-scp.auto.html (19 Sept. 2004).

Boran, Seán. "All About SSH - Part I/II." 22 July, 2004. URL: http://www.boran.com/security/sp/ssh-part1.html#Features (10 Sept. 2004).

Enterprise Distributed Technologies. "edtFTPj/SSL FAQ – Answers" 2004 URL: http://www.enterprisedt.com/products/edtftpjssl/faq-answers.html (15 Sept 2004).

Ford-Hutchinson, Paul. "ftps - FTP-SSL and FTP-TLS - the state of play" 13th Sep 2004. URL: http://www.ford-hutchinson.com/~fh-1-pfh/ftps-ext.html . (19 Sept 2004).

Manek, Joe. "Securing FTP. Then and now." 15 Dec. 2003. URL: http://www.sans.org/rr/papers/index.php?id=1462 (10 Sept. 2004).

Open SSH. 29 Aug. 2004. URL: http://www.openssh.com/. (15 Sept. 2004).

SANS Institute. "The SANS Top 20 Internet Security Vulnerabilities" Version 4.0. 8 October 2003. URL: http://www.sans.org/top20/ (15 Sept. 2004).

Symark. "Securing Multiple Unix Operating Systems with Symark PowerPassword® and Symark PowerBroker®." 2002. URL: http://www.cherryweb.tv/symark/WhitePapers/wp_securing_os_with_pb_and_pp.pdf (15 Sept. 2004).

Tobias, Steve. "Case Study in Secure File Transfer: Implementing Secure FTP with SSL In a Healthcare Organization." 14 July 2004. URL: http://www.sans.org/rr/papers/index.php?id=1462. (10 Sept. 2004).

University of Kentucky, Center for Computational Sciences. "Secure file Transfer." URL: http://www.ccs.uky.edu/machines/sftp.html (10 Sept. 2004).

Yamamoto, Hirotaka. "SSH2 Quick Start." 22 Sept. 1998. URL: http://www.yl.is.s.u-tokyo.ac.jp/~ymmt/ssh2.html (15 Sept. 2004).

Bibliography

Northcutt, Steve et al. (2003). Inside Network Perimeter Security. Indianapolis, IN: New Riders.

Viega J., Messier M., Chandra P. (2002). Network Security with OpenSSL. Sebastopol, CA: O'Reilly.

Hosner, Charlie. "OpenVPN and the SSL VPN Revolution." 8 Aug. 2004. URL: http://www.sans.org/rr/papers/index.php?id=1459. (18 Sept. 2004).