


Realtime
publishers

"Leading the Conversation"

The Shortcut Guide[™] To



Securing Automated File Transfers

sponsored by



Ed Tittel

Introduction to Realtimepublishers

by Don Jones, Series Editor

For several years, now, Realtime has produced dozens and dozens of high-quality books that just happen to be delivered in electronic format—at no cost to you, the reader. We’ve made this unique publishing model work through the generous support and cooperation of our sponsors, who agree to bear each book’s production expenses for the benefit of our readers.

Although we’ve always offered our publications to you for free, don’t think for a moment that quality is anything less than our top priority. My job is to make sure that our books are as good as—and in most cases better than—any printed book that would cost you \$40 or more. Our electronic publishing model offers several advantages over printed books: You receive chapters literally as fast as our authors produce them (hence the “realtime” aspect of our model), and we can update chapters to reflect the latest changes in technology.

I want to point out that our books are by no means paid advertisements or white papers. We’re an independent publishing company, and an important aspect of my job is to make sure that our authors are free to voice their expertise and opinions without reservation or restriction. We maintain complete editorial control of our publications, and I’m proud that we’ve produced so many quality books over the past years.

I want to extend an invitation to visit us at <http://nexus.realtimepublishers.com>, especially if you’ve received this publication from a friend or colleague. We have a wide variety of additional books on a range of topics, and you’re sure to find something that’s of interest to you—and it won’t cost you a thing. We hope you’ll continue to come to Realtime for your educational needs far into the future.

Until then, enjoy.

Don Jones

Introduction to Realtimerepublishers..... i

Chapter 1: File Transfer Security Issues.....1

FTP: The 10,000 Foot View2

 How FTP Is Used.....2

 FTP Behavior3

Where Is FTP Used?.....7

What Makes FTP Insecure.....8

 Passwords and Contents in Plain Text8

 Lack of High-Level Integrity Checks9

 Separate Channels and Random Port Allocation.....10

 Abuse of Built-In Proxy Features10

Other Insecure Transfer Methods and Models.....10

Understanding the Role of Automation12

 Scheduled Tasks.....12

 Command-Line FTP and OS-Level Schedulers13

 Centralized Job Scheduling Solutions13

 Built-In Application Capabilities.....13

 Use of External Tools and APIs.....13

 Use of Network Drives13

File Transfer and Regulatory Compliance.....14

 Privacy and Confidentiality Issues.....14

 Relation to SOX, HIPAA, PCI, and Other Compliance Regimes15

Summary16

Chapter 2: Planning Secure File Transfer Deployment17

Implementing FTP Inventory.....18

 Capture and Characterize Network Traffic by Location.....20

 Perform a Port and Protocol Inventory21

 Trace Items of Potential Interest.....21

 Track Unknown Items Back to Their Original Locations22

 Identify All Types of File Transfer.....22

Perform a Platform Inventory23

 Security Documentation.....23

 Compliance Management23

Identify Platforms and OSs in Use.....	24
Identify Secure File Transfer Solutions	25
Look for Intersection Across Platforms.....	27
Communications Protocols	27
Software Interoperability	28
Data and Information	28
Making a Technology Selection: Performance Issues	29
Costs of Encryption and Compression.....	30
Identifying and Minimizing Potential Bottlenecks	31
Benefits of Hardware Acceleration.....	32
Making a Technology Selection: Security	32
Key Management Schemes.....	32
Public Key vs. Private Key Infrastructures.....	33
Benefits of Centralized Key Management.....	33
PKI and Interoperability Issues.....	34
Avoid Security Through Obscurity.....	34
Compliance and Security Audits	36
Stick to Security Standards	38
Making a Technology Selection: File Transfer Topology	38
Hub-and-Spoke Transfer Environment: Pros and Cons.....	39
Finding the Best Fit for Your Environment.....	40
Reworking Automation.....	40
Scripting, Scheduling, and Batch Processing	40
Summary	41
Chapter 3: Securing File Transfer.....	42
TCP/IP and Network Reference Architectures	42
What Is Tunneling?.....	45
What Is IPsec?.....	46
Why Is IPsec Used?	46
How Is IPsec Used?	47
What Is Secure Socket Layer?	50
Why Is SSL Used?	51
How Is SSL Used?	51

What Is TLS?	52
Why Is TLS Used?.....	52
How Is TLS Used?.....	53
What Is SSH?.....	53
How Is SSH Used?.....	53
Using VPNs	54
IPsec VPNs	54
Implementation Approaches	55
SSL VPNs.....	55
Implementation Approaches	56
SSH VPNs.....	56
Implementation Approaches	56
Other VPNs.....	57
Securing FTP	57
SSH.....	57
SFTP	58
SCP (Secure Copy).....	58
FTPS (aka FTP/SSL)	59
Other Approaches	59
Tunneling with SSH.....	60
The Push/Pull Concept.....	60
Automated File Transfers	61
Keyed Logins for Transfer (No Password).....	62
Keyed Logins for Transfer (With Password).....	63
Securing Automated Transfers	63
Preferred Authentication Methods.....	64
Using Public-Key Mechanisms (With and Without Certificates).....	64
Avoid Embedding Passwords in Scripts or Communications	65
Avoid Weak Passwords	65
Summary.....	66
Chapter 4: Compare/Contrast SFTP, FTPS, and IPsec.....	68
Host-to-Host VPNs	69
Host-to-Gateway VPNs	69

Gateway-to-Host VPNs	69
Gateway-to-Gateway VPNs.....	69
Software Solutions	70
Hardware Solutions.....	70
Intranet Solutions	71
Extranet Solutions.....	71
Tunneling Protocol Options.....	72
Advantages.....	72
Advantages of IPsec.....	72
Advantages of SSL	73
Advantages of SSH.....	73
Disadvantages	74
Disadvantages of IPsec	74
Disadvantages of SSL.....	74
Disadvantages of SSH.....	75
Security Considerations	76
Strong Encryption and Authentication Using Standard Algorithms.....	76
Consider How Control and Data Channels May Be Handled.....	77
Security Policy and Compliance Requirements.....	77
When to Use SFTP.....	78
When to Use IPsec	78
When to Use FTPS.....	79
Security	79
Flexibility	79
Certificates and Certificate Authorities	79
The Demilitarized Zone	80
FTP Placement Inside the Firewall.....	80
Choosing a File Transfer Solution	81
Total Cost of Ownership.....	81
Assessing Overall Applicability	82
Summary	83

Copyright Statement

© 2007 Realtimepublishers.com, Inc. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtimepublishers.com, Inc. (the "Materials") and this site and any such Materials are protected by international copyright and trademark laws.

THE MATERIALS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. The Materials are subject to change without notice and do not represent a commitment on the part of Realtimepublishers.com, Inc or its web site sponsors. In no event shall Realtimepublishers.com, Inc. or its web site sponsors be held liable for technical or editorial errors or omissions contained in the Materials, including without limitation, for any direct, indirect, incidental, special, exemplary or consequential damages whatsoever resulting from the use of any information contained in the Materials.

The Materials (including but not limited to the text, images, audio, and/or video) may not be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, in whole or in part, except that one copy may be downloaded for your personal, non-commercial use on a single computer. In connection with such use, you may not modify or obscure any copyright or other proprietary notice.

The Materials may contain trademarks, services marks and logos that are the property of third parties. You are not permitted to use these trademarks, services marks or logos without prior written consent of such third parties.

Realtimepublishers.com and the Realtimepublishers logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners.

If you have any questions about these terms, or if you would like information about licensing materials from Realtimepublishers.com, please contact us via e-mail at info@realtimepublishers.com.

[**Editor's Note:** This eBook was downloaded from Realtime Nexus—The Digital Library. All leading technology guides from Realtimepublishers can be found at <http://nexus.realtimepublishers.com>.]

Chapter 1: File Transfer Security Issues

Since the creation of the first computer network, there has been sustained interest in leveraging network media to transport data. Just as interstate highways do for commercial transportation, modern computer networks serve as a primary conduit through which business is conducted, so that (virtual) goods, services, and information can be expedited to any location around the world.

The dawn of the Internet era brought with it innumerable opportunities for enterprising and pioneering engineers eagerly seeking to exploit this medium as a means to exchange information. Among these early-information sharing services you can find many forms of information and file sharing protocols and software, including early Usenet and UNIX-to-UNIX Copy (UUCP) implementations to the aptly-named File Transfer Protocol (FTP) that remains widely used for file transfer to this very day. These two approaches illuminate early efforts at distributing information on a wide scale. Although UUCP leverages less server-side capability and tends toward client-to-client (or peer-to-peer) behavior, FTP operates squarely within the framework of a client/server model. Both typify two major classes of file sharing protocols and architectures, with many variations on the same theme still in active development and current use today.

FTP is an official protocol specification that details precisely what is involved in establishing, maintaining, and tearing-down reliable, connection-oriented Transmission Control Protocol/Internet Protocol (TCP/IP) communication. As an Internet Engineering Task Force (IETF) standard, FTP works within just about any conceivable combination of standard FTP client and server protocols—in other words, there are no vendor lock-ins, no proprietary protocol dependencies, nor any real need to ensure version or build compatibility between FTP clients and servers. Owing to its longstanding appearance and widespread use, FTP is an insecure means of communication and file or data transfer. This chapter explores this reality, then goes on to describe similar protocol families as points of comparison.




The IETF develops and promotes Internet standards especially with regard to TCP/IP. The primary criteria for meeting IETF standards is that a given protocol, service, or technology is interoperable among various other TCP/IP products.

Its built-in vendor neutrality has gone a long way in extending the lifespan of FTP, which remains a vital component even in today's increasingly security-aware small to medium business and larger enterprise network infrastructures. Consider this a vindication of the work of early internetwork designers. What remained unforeseen at the time of FTP's design and original implementation, and therefore was not factored into those efforts, was the growing tide of abuse or misuse of Internet resources by unscrupulous end users or outsiders with crooked (or at least, questionable) agendas.

During the formative years of the Internet (between 1960 and 1980), designers trusted users to utilize network resources safely and responsibly, without reckless abandon or malicious intent. Accounting for most users was also easy: network computing resources were initially so scarce that access was allotted in time slices so that everyone with access perforce shared that access among others. However, in today's world of multi-user domains that can encompass geographically dispersed locations, this sort of blind trust is no longer feasible.

For more information about the history of FTP, version 1 is characterized first and foremost in Request for Comments (RFC) document 114 (RFC 114) issued on April 10, 1971. Subsequent RFCs examine other related subjects such as File Transfer and Recovery (RFC 133), Comments on RFC 114 (RFC 141), and Data Transfer Protocols (RFC163). In these documents, you'll find the primitive outlines for what was to become the most heavily relied-upon all-purpose file transfer protocol until the advent of more advanced peer-to-peer networking infrastructures began to occur in the 1990s. Since that time, several major revisions (up to version 5.1) detail the last major developments to the FTP framework, some of which go on to address much-needed features and functions, including Firewall-Friendly FTP (RFC 1579) and FTP Security Extensions (FTP 2228), among others.

 For a comprehensive list of FTP-related and similar RFCs see <http://www.wu-ftpd.org/rfc>.

FTP: The 10,000 Foot View

Let's begin with a high-level overview of FTP and briefly examine certain low-level details that underscore inherent weaknesses in its existing design. The discussion begins with an explanation of how, where, and when FTP may be used in a business environment. The focus then turns to the FTP login process, which is illustrated with packet traces taken from an open source protocol analyzer to depict how this protocol blatantly exposes logon credentials "on the wire" while the packets that contain that information are in flight from sender to receiver.

How FTP Is Used

FTP may be used any time data needs to be transferred across a TCP/IP network. Though most readers may immediately and exclusively associate FTP use with end-user initiated file transfers between desktops and FTP servers, it may also be invoked in other circumstances, too. Careful observation and analysis of TCP applications within enterprise settings often reveal scripts, batch files, automation software, or even applications that leverage internal FTP libraries or external binaries to transmit end-user data, database information, DNS zone maps, Web server content, software updates, source code packages or patches, and client or server-side upgrades. In a nutshell, there is very often a great deal more FTP-related activity going on within even the best-run networks than many network professionals may know about or recognize.

File transfers may occur between two machines sitting side by side, between machines separated only by offices or floors in the same building, between geographically separated remote offices within a single enterprise, or across the network boundaries between two independent organizations. Such concentric rings of activity may also experience significant overlap, in that offsite data may be pushed to central distribution servers and then shipped off to e-vault operators for safekeeping or to meet an organization's backup and restore policy requirements. In a great many instances, FTP is the underlying mechanism that drives such automated processes and serves as a general vehicle for business information exchange. Hopefully, this evocative and entirely typical illustration may help you understand why security can become geometrically more difficult to assess, define, and implement as the target network topology increases in scope and scale, and by the number of independent parties involved.

FTP Behavior

By default, most FTP servers listen on port 21/tcp unless configured explicitly to listen on a different port number. A client connection to this port initiates a sequence of events that gets the data transfer process underway by first creating a control stream through which FTP commands pass between client and server. Actual data transfer occurs in a separate data stream, which may be further characterized by passive- or active-mode behavior.


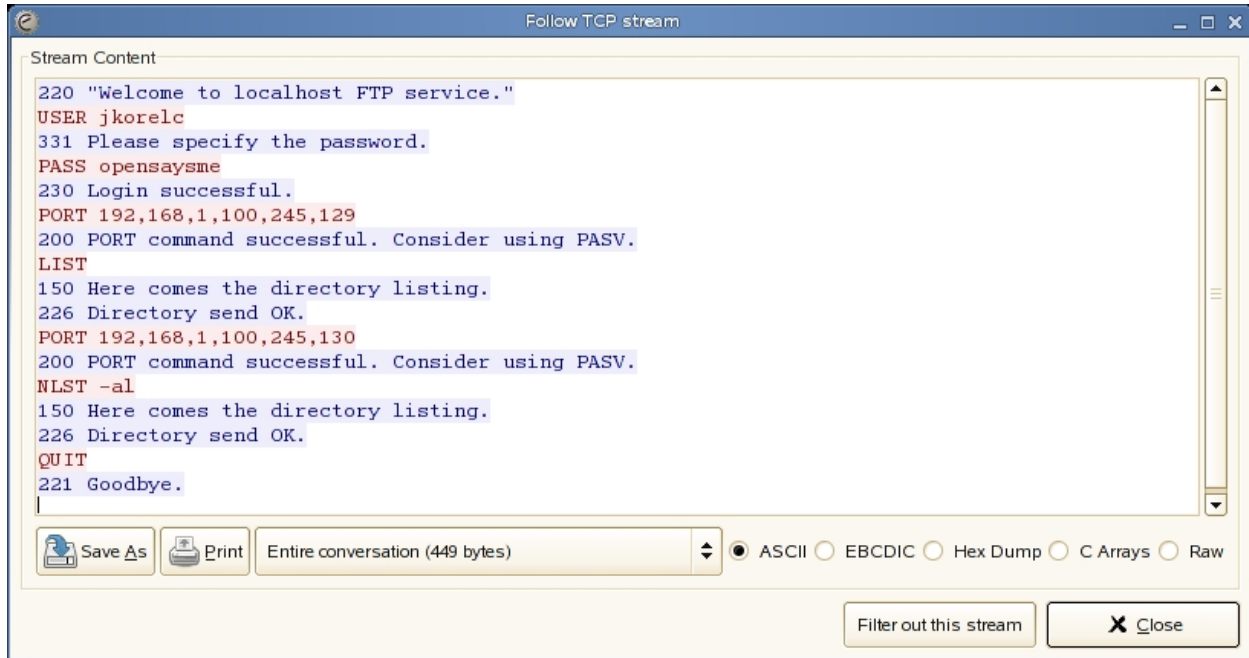
 In active mode, the client opens a random port (> 1023), sends the server the random port number on which it is listening over the control stream, and waits for a connection from the server. When the server initiates the data connection to the client, it binds the source port to port 20 on the server. In passive mode, the server opens a random port (> 1023), sends the client the port on which it is listening over the control stream, and waits for a connection from the client. In this case, the client binds the source port of the connection to a random port greater than 1023.

Figure 1.1 illustrates an FTP transaction conducted over the network in the open, where a network protocol analyzer can clearly pick up logon credentials (among other crucial details).



```

Stream Content
220 "Welcome to localhost FTP service."
USER jkorelc
331 Please specify the password.
PASS opensaysme
230 Login successful.
PORT 192,168,1,100,245,129
200 PORT command successful. Consider using PASV.
LIST
150 Here comes the directory listing.
226 Directory send OK.
PORT 192,168,1,100,245,130
200 PORT command successful. Consider using PASV.
NLST -al
150 Here comes the directory listing.
226 Directory send OK.
QUIT
221 Goodbye.

```

Save As Print Entire conversation (449 bytes) ASCII EBCDIC Hex Dump C Arrays Raw

Filter out this stream Close

Figure 1.1: An example of FTP operating in clear text, as shown through a network protocol analyzer.

Compare and contrast this information with the data captured in Figure 1.2, which is a secure method of FTP that is much less revealing.

```

SSH-2.0-OpenSSH_4.5
SSH-2.0-5.1.3.8 SSH Secure Shell
.....#:...M...RS.U...2diffie-hellman-group1-sha1,extension1-sha1@ssh.com...ssh-dss,ssh-
rsa...Pcrypticore128@ssh.com,aes128-cbc,aes192-cbc,aes256-cbc,3des-cbc,seed-
cbc@ssh.com...Pcrypticore128@ssh.com,aes128-cbc,aes192-cbc,aes256-cbc,3des-cbc,seed-
cbc@ssh.com...)crypticore-mac@ssh.com,hmac-md5,hmac-sha1...)crypticore-mac@ssh.com,hmac-
md5,hmac-sha1...none,zlib...none,zlib.....*...d.
....
'....A.4...d...~diffie-hellman-group-exchange-sha256,diffie-hellman-group-exchange-
sha1,diffie-hellman-group14-sha1,diffie-hellman-group1-sha1...ssh-rsa,ssh-dss...aes128-
cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour128,arcfour256,arcfour,aes192-cbc,aes256-
cbc,rijndael-cbc@lysator.liu.se,aes128-ctr,aes192-ctr,aes256-ctr...aes128-cbc,3des-
cbc,blowfish-cbc,cast128-cbc,arcfour128,arcfour256,arcfour,aes192-cbc,aes256-cbc,rijndael-
cbc@lysator.liu.se,aes128-ctr,aes192-ctr,aes256-ctr...U hmac-md5,hmac-sha1,hmac-
ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-md5-96...U hmac-md5,hmac-sha1,hmac-
ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-
md5-96...none,zlib@openssh.com...none,zlib@openssh.com.....Z
$.b%...Iq..T%..K...MN.H.P5tcY...../g...U...
...ON.H.CC...2:..O.....o...".@..w7...f.NK...-.8.
%.....SJ;l...>.u.....'.....d...|.....ssh-dss.....WX.r4tt4....%p4
{.d...7f.:.i&w....D....}...s...}...c..B..
.....(P.
?h.n.A.....8S.F.'1.....y6..%
<RA...oA..g..LO...w.....e.../4..7..Z..o.....F...^h..Z.....S:I.....1..K...g3!
Zg.g...IK..n.ld.z....P8...o...:..N..{$.....~..M.....>..rj%...
m02...(...c)H.h...{.9...t...*...!...L.....".....L...$.....='Xg.Ju..
+3.kz~.z..6y...1%.....6...{.....*.....w.....C.
\>...0vzAX..8.....}.....}.....$.H...t|m...:K.....}.U3R.]
C..l3.x..J..8k...Y...i".y.R6.g.f_6..n...0q...uzr.....3.s...~....@c...
\.....*.....7....ssh-dss...(...s.....n.....}. ..../.H.h..X.xc...e.....
.....

```

Figure 1.2: An example of Secure FTP, which is much less informative about its transaction.

When operating in active mode, the FTP client will open and bind to a local TCP port number greater than 1023 (port number ranges vary by implementation), listen for a connection, then issue a command to prompt the FTP server to dial-in to the port that the client advertises in its connection request. In turn, the FTP server binds locally to port number 20, initiates a data connection back to the client on the specified listening port, and begins transmitting data. Figure 1.3 provides a flowchart depicting this behavior.

Active FTP Connection

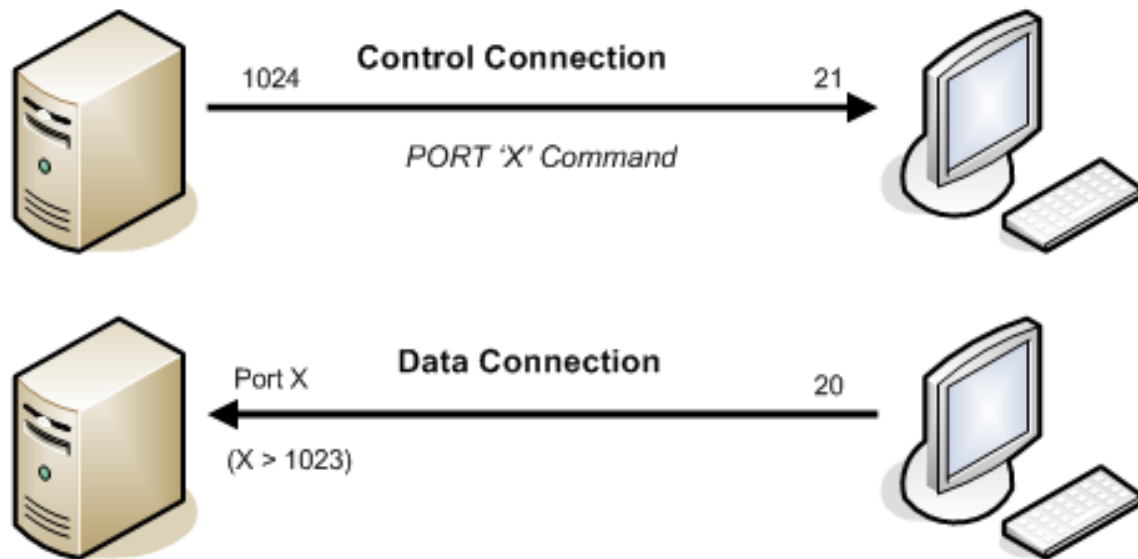


Figure 1.3: The active FTP transfer method.

When operating in passive mode, the server instead opens and binds to a local TCP port number greater than 1023, then waits for the client to connect a data stream to that specified server port. Figure 1.4 offers an example of a passive-mode transfer method.

Passive FTP Connection

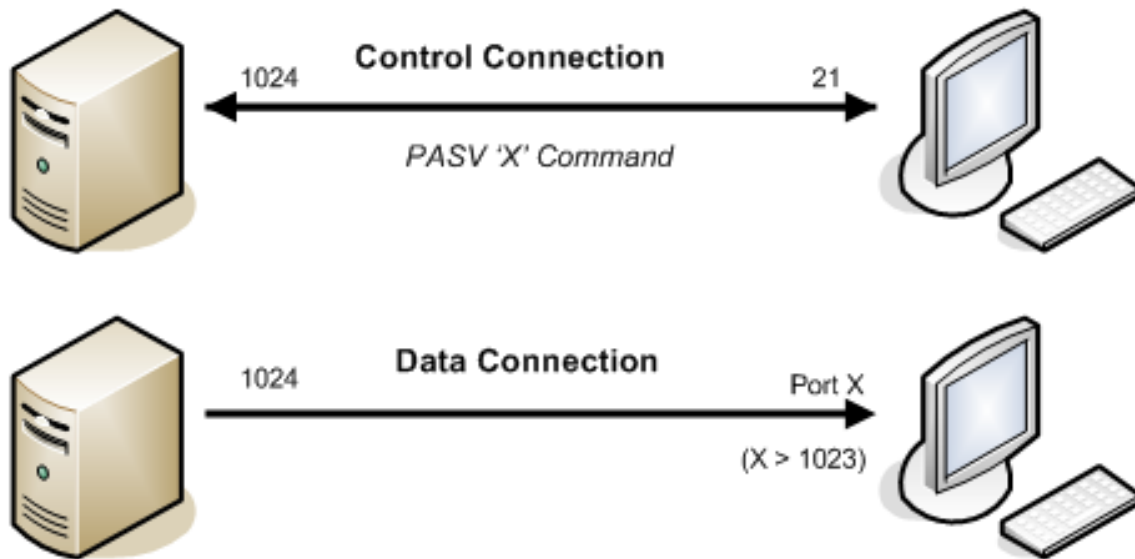


Figure 1.4: The passive FTP transfer method.

In addition, an FTP server may also provide anonymous access to users, usually with read-only permissions for most files and directories, but sometimes with a writable directory to enable anonymous clients to upload content to that server. Either way, FTP insulates its users from platform differences so that any FTP client needs to know no specifics about the FTP server platform to exchange files with it. One of the best explanations for the enduring popularity and continued use for FTP is that it makes no difference whether data is exchanged between a UNIX server and a Windows workstation, or a Linux server and an Apple notebook, or even an IBM mainframe and a SunOS workstation—FTP enables ready, simple transfer between clients and servers without much muss or fuss.

Where Is FTP Used?

A short, succinct, but ultimately uninformative answer to this admittedly leading question might be “Everywhere.” In fact, FTP’s ubiquity can be attributed to its long-standing status as an Internet standard, which in turn explains the multitude of operating system (OS) platforms that include some basic form of FTP client software (and occasionally server software) as part of their core distributions. It also explains why so very many shareware, freeware, and commercial versions of this software are also available for just about any computing platform you might name. Equally important, nearly all versions of Windows (including Vista, the newest member of the Windows OS family) contain a command-line FTP client, while UNIX and Linux platforms include both client and server packages within their standard distribution sets. Basically, FTP is used anywhere and everywhere—you needn’t look long or hard to find it in use on a computer in your immediate vicinity, wherever that might happen to be.

The following list highlights examples of situations in which you’re likely to find FTP at work in your own organization:

- End-user applications—Client FTP components may be standalone programs or modular components—that is, FTP client software may run as a separate utility or might simply operate as an integrated library within some larger application with file-sharing capabilities. FTP servers generally follow a specific RFC-defined pattern of behavior (unless custom onsite modifications are made to the server software), so such traffic is easy to spot on the network once you begin to search for its presence.
- Intra-enterprise file transfers—Many bulk or high-volume file transfer tools used in enterprise IT operations also tend to incorporate some form of FTP at their cores, some implementations of which may not even be fully RFC-compliant. Site-to-site transactions may utilize other, non-FTP forms of file transfer based on proprietary protocols developed in-house and implemented on-site, but these, too, sometimes operate around an FTP core.
- Hard-coded into proprietary or open source applications—Proprietary back-up software may utilize modified forms of FTP implementing vendor-specific optimizations or localizations to best serve specific needs in their file-sharing or exchange software. Open source programs may incorporate FTP capabilities within other software packages through extensible plug-in frameworks or shared library components.

Large-scale data movers often utilize basic forms of FTP behind the scenes to provide a transparent conduit for bulk file transfers. FTP command-line clients reside on many desktops and workstations, where third-party software additions sometimes introduce other entry points for file exchange as well. Another ongoing phenomenon is the integration of FTP clients into Web browsers to deliver a single software tool that adds file transfer to its repertoire of other activities (Web page access, news reader, email client, and so forth).

What Makes FTP Insecure

What could possibly be wrong with such a friendly, open, and practically omnipresent standard? Here again, the concise answer is “Security,” but the brevity of that response fails to do full justice to the gravity of the potential exposures involved. To clarify this perhaps too-cryptic utterance, let’s say that FTP is inherently insecure for numerous reasons, some of which include:

- Passwords and file contents are transferred in clear, unencrypted text
- No integrity checks are performed on the receiver
- Separate control/data connections require additional firewall logic for accounting purposes
- Active client-side connections are difficult to filter due to arbitrary port allocation on receiver
- Abuse of built-in protocol proxy features (PORT command) enable bounce scans to other hosts

Passwords and Contents in Plain Text

First and foremost in any security analyst’s mind when data moves across a network is the degree to which such communication is visible to third parties, the vast majority of which may be safely assumed to be unauthorized to view or touch that information. In this kind of situation, in fact, security mavens take particular care in the handling of credentials needed to access resources—namely, the account names and passwords typically required to establish an identity and establish access rights, plus the actual contents of the commands and files that pass between clients and servers. The prevailing best practices for protecting account and password information, or other data used to establish identity (such as certificates, keys, tokens, and so forth), are to impose extremely high degrees of encryption on such traffic and to raise the bar high enough for those who might seek to crack such encryption to make it more troublesome and time consuming than the data it protects is worth. Often these high-security data exchanges also provide an effective conduit for session-specific shared encryption keys so that file contents may be protected somewhat less aggressively but nevertheless kept safe enough from prying eyes and unauthorized access to be useful.

Alas, neither the FTP standard nor standard-compliant implementations require or provide any such protection for the account and password information passed from client to server during the login and session start-up process or for the data and control message exchanges that occur between client and server once an FTP connection has been established between them. This is effectively like broadcasting every transmission as a public announcement to anyone within listening range. Because FTP transmissions (referred to here as plain text or “in the clear”) are exposed on the network and visible to virtually anyone, it doesn’t matter what security implementations may be in place on the server. A savvy attacker with sufficient network access and basic protocol analysis knowledge can eavesdrop on an FTP connection initiation, obtain a set of valid logon and password credentials, then snoop on or siphon off any data being transferred. Already, this attacker has logon credentials, which could be tried against other servers and services to leverage access in other areas of the network or capture and leak potentially sensitive or confidential information being exchanged. From a security standpoint, unadulterated FTP use is a penetration waiting to happen, or an exploit in the offing.

 FTP control channels are left exposed to network protocol analyzers.

Lack of High-Level Integrity Checks

In addition to the clear-text communication problem, there are no receiver-side integrity checks against transmitted data, other than those that inhere to the TCP segments that comprise client-to-server or server-to-client file exchanges. This leaves open yet another avenue for attack because an attacker needn’t obtain logon credentials at all yet can still violate an FTP exchange. From a Man-in-the-Middle (MitM) position, a hypothetical attacker could conceivably intercept a targeted file for transfer en route, say an important firmware upgrade pushed via Trivial FTP (TFTP) or software update archive, then send a modified version on to the receiver. As long as this unknown intermediary maintains the right level of integrity data at the TCP level while replacing the data in motion with other, possibly compromised or vulnerable data, integrity becomes moot. The MD5 sums that accompany most software packages will be rendered useless because a modified checksum can be injected into the same stream, and only due diligence—or higher-level integrity checks—can expose such nefarious activities or insertions.

Without in-line integrity checks against what is sent versus what is received, there can be no confidence in the validity of transferred data. Can you be certain that a source code archive does not contain a patch to reduce, eliminate, or corrupt some or all the security protections it purportedly implements? Are you positive there is not a Trojan, backdoor, or accompanying malware present in such code? The sense of caution necessary must be proportional to the specific nature, environment, or exposures unique to each business infrastructure—but such caution should always remain a valid concern for any network professional. For what it’s worth, this also explains why so many vendors include specific signature mechanisms as part of their download and update architectures—simply because such mechanisms supply the higher-level integrity checks necessary to make sure that what the recipient obtains matches what the sender transmitted precisely and exactly.

Separate Channels and Random Port Allocation

Many workarounds for these kinds of shortcomings—such as the use of restrictive firewall Access Control Lists (ACLs) and accompanying rule sets—are not without their own unique challenges. FTP may be characterized as a high-latency protocol with lots of chatter during initial connection setups that operates in two distinct modes: passive and active. During active mode, the FTP server prods the client to open a listening port to receive data and, once connected, can perform the same manipulations as if it had connected to the server. The good news is that all such ports are allocated above the well-known TCP and UDP port numbers (1024 and up), which you may safely eliminate from a firewall filter. The bad news is that the range from 1024 to 65535 must be examined, and filtered, when a search for active FTP connections is positive or whenever additional software components are used to track such sessions. Passive-mode applications do not present these same challenges for a network administrator and are much easier to monitor because the FTP server is responsible for establishing a data port for the client to utilize.

Abuse of Built-In Proxy Features

The last item on this list of known FTP exposures is the FTP bounce attack. An FTP bounce or proxy attack makes deliberate misuse of the PORT command, whereby an attacker leverages a server into acting as a go-between for network reconnaissance scans on the attacker's behalf. This technique makes attack scan patterns appear to originate from an abused FTP server instead of the actual attacker's location, which may or may not be its true point of origin anyway.

Imagine the problems that an FTP bounce attack can create for business-to-business data exchange, where proprietary or confidential information is governed by mutual Non-Disclosure Agreements (NDAs) and requirements for high-levels of secrecy and confidentiality. Such an attack can not only strain external trust relationships but also damage internal relationships.



If your FTP is in the DMZ but has access to other systems inside the internal network, an attacker may be able to map an internal network topology through an unsuspecting (or unprotected) source.


Other Insecure Transfer Methods and Models

While we are on the subject of clear-text protocols and file transfer methods, FTP is not the only culprit of this kind. Thus, the following observation must be elucidated: secure implementation is generally an afterthought in the software production process because it's time consuming, complex, and difficult to get right the first time. Because we humans can only logically deduce so much about application behavior, entire categories and classes of vulnerability cannot be accounted for up front either easily or straightforwardly. Creating procedures and standards can address this. Parts of the design phase must introduce further angles on security and secure implementations, including investigation of third-party or recycled software libraries that may very well contain programmatic or configuration errors leading to exposures or vulnerability, if they are not guilty of eschewing security mechanisms altogether, as is the case for FTP.

In the same vein, other long-established protocols and services are susceptible to many of the same or similar attacks as is FTP. The list of “suspects” includes the following in this case:


- File Exchange Protocol (FXP)—A method of transfer that uses FTP between servers without routing through a client connection. Whereas typical FTP sessions involve client/server pairings, with FXP, two server endpoints are controlled by a single client connection but data is exchanged between a pair of servers.
- Network File System (NFS)—A distributed file system protocol that establishes remote exports for client callers, typically characterized by UDP traffic that transmits transactional information in the clear. An extension of NFS called WebNFS enables easy integration with Web browser clients and operates across firewalls with relative impunity.
- Network Information Service (NIS)—Acts as a central authentication hub for managing large-scale NFS exports (thus, is UNIX-specific) in a network topology. An NIS server (usually denoted with a plus, NIS+) maintains state information and grants access to objects for subjects that successfully authenticate through its identity checks.
- WebDAV—An acronym that expands to Web-based Distributed Authoring and Versioning, which refers to both an IETF working group and the set of HTTP extensions with the goal of making online content both readable and writable with greater transparency to end users. The WebDAV protocol makes remote Web server content manipulation easier and provides a general-purpose online file-storage accessible from anywhere.
- Common Internet File System (CIFS)/Server Message Block (SMB)—SMB is an application-level network protocol (like WebDAV) that provides shared access to remote resources. Chief among these resources are public exports of client and server shares (much like NFS exports). There are many re-implementations of SMB for both UNIX and Windows platforms; most notably, the free Samba.
- rsync—Stands for remote synchronization, a timeless tool for maintaining remote synchrony among separate endpoints. This may be simply shoring up local files with the most current remote versions, as in the case of obtaining the latest software updates for a UNIX platform or package, or managing uniform distribution of changes among the servers of an entire organization.
- Peer-to-Peer (P2P) applications—The decentralized file-sharing standard used around the world to provide a working alternative to the typical client/server network paradigm. Nodes are connected largely in an ad hoc fashion and most processing is done client-side; thus, detecting such traffic is not particularly challenging. Common P2P networks, applications, and protocols include such well-known names as Usenet, Napster, OpenNAP, Gnutella, and Freenet.

As is the case for some of the preceding protocols and protocol extensions, some security measures have been set in place to enforce authorized behavior among clients and the services with which they interact, but other protocols and services have been left largely or completely unsecured. Even when security measures are in place, the protection they afford may either be insufficient to thwart intrusion or may themselves introduce other vulnerabilities.

 FXP clients are designed to support secure data channels through the Secure Socket Layer (SSL) or Transport Layer Security (TLS) by using other FTP security extensions such as CPSV or SSCN. However, by enabling this support, a server may become susceptible to FTP bounce scans.

Furthermore, the rare CPSV and SSCN extensions themselves are susceptible to MitM attacks whenever servers fail to verify the SSL certificates used during any data exchanges.

In the world of medicine, only a crackpot doctor insists on treating symptoms instead of causes. Your focus as a network professional can hardly differ. As a purveyor of network-wide security solutions and general well-being, the health of your subjects and objects must be a top priority. Treating an affected protocol such as FTP with patchwork extensions such as FXP only hints at the fact that FTP is inherently insecure by design. It really requires a major overhaul to improve its baseline security. Such treatment does not necessarily address root causes or offer a completely secure replacement for an insecure toolset.

 Subsequent chapters will dig into the details involved in securing FTP and similar transfer. Possible approaches range from preserving existing clear-text transfer protocol methods within the capable embrace of more robust network security protocol frameworks (such as virtual private networks—VPNs) to switching over to more suitable service replacements designed with security as well as service in mind (such as Secure FTP, commonly known as SFTP).

Understanding the Role of Automation

Automation is the key to managing any large-scale IT infrastructure. There are only so many routines a diligent administration staff can handle at any given moment, which explains why it's so important to leverage business-class servers when it comes to routine, repetitive, and mission-critical tasks. Automated file transfers are integral to daily business operations, where such tasks normally include the import and export of large databases or codebases between internal and external sources, backups, data store, direct replication, and so forth. In this kind of environment, batch or scheduled jobs of all kinds occur at regular intervals and are on demand on daily, weekly, monthly, quarterly, or other periods to help keep the data moving, protected, and to accommodate regular business or organizational activities.

Scheduled Tasks

Scheduled tasks refers to file transfer communications and capabilities that manifest themselves in any number of ways or that use any of a number of tools along with any number of network connections to other clients and server services. Such tasks may be entirely automated, such as routine internal client workstation or database backup processes, server backups, database or directory backup or replication, or end-user workstations resuming previously interrupted transfer sessions upon restoration of necessary services or connections. Key areas that involve file transfer appear in the sections that follow, in no particular order.

Command-Line FTP and OS-Level Schedulers

Simple command-line FTP clients are easy to use, easily scripted, and leverage a huge amount of power in terms of data transfer. Naturally, an FTP client lends itself to light administrative duties in the form of automated file backups and exchanges at specified intervals throughout the work day, week, and year. On Windows derivatives, the Task Scheduler commonly handles the timely execution of batch jobs, and for UNIX and Linux hosts, either *at* or *cron* command-line utilities serve the same purpose.

Centralized Job Scheduling Solutions

For medium to large business and enterprise networks, and for networked storage systems at all levels, automation plays a key role in maintaining routine workloads and assigned tasks. Automation is also a key to backup, archival, and meeting compliance requirements for records retention in many industries and throughout the world as required by various country data protection laws.

Built-In Application Capabilities

Web browser interfaces and server-to-server file exchanges illustrate client and server software that incorporate some form of file transfer capability. There are many other examples of this kind of activity, such as network file systems, storage systems, or other forms of file interaction that also involve network access.

Use of External Tools and APIs

The UNIX *rsync* utility and the general-purpose WebDAV extensions define two well-known situations in which file transfers occur and users may not necessarily be aware of what is going on in the background. Furthermore, such file transfers may often occur in the clear, enabling eavesdroppers to intercept and possibly intervene in such transactions.

Use of Network Drives

Network drive mappings are common on many UNIX, Linux, and Windows machines alike, from SMB/CIFS shares to NFS/NIS exports. FTP is a natural choice when it comes to push-and-shove because it is found on nearly every platform and is compatible across so many target hosts.


File Transfer and Regulatory Compliance

Thanks to the Health Insurance Portability and Accountability Act (HIPAA) of 1996, health care organizations must handle sensitive patient information with utmost care and do their utmost to prevent and avoid unauthorized access or disclosure. In turn, this means that careful thought and thorough consideration to storage, transmission, and disclosure of protected health information (PHI) must be performed under regulations that are enforced by federal law. In the same general vein, the Graham-Leach-Bliley Act put the onus on many types of financial institutions to keep careful control over nonpublic personal information (NPI) as well. Credit card companies have their own requirements as well for the safekeeping and safe handling of sensitive customer data including processing, transmission, and retention.

Privacy and Confidentiality Issues

You have undoubtedly read numerous public announcements, in full-blown press releases, from various businesses, organizations, agencies, or universities that disclose security breaches and/or leakage or loss of sensitive information regarding their customer, client, or student and faculty databases. Such things do happen, and in these unfortunate circumstances, they happen on a large and publicly visible scale. You hear about these things not merely because news agencies report on them, but because standards and compliance controls make these entities responsible for reporting on such events to assign financial responsibility and to document infrastructure security and related problems or issues. One inescapable truth about anyone's role or function within any organization that is governed by such laws and regulations is that individuals and organizations may be held responsible—and in some cases criminally or financially liable—when things go terribly wrong.

Storing confidential electronic information securely is the primary responsibility of the Chief Information Officer (CIO) and the IT administration staff assigned to maintain storage facilities. By extension, other employees and contractors (including both internal and external systems auditors) must also maintain professional integrity and handle confidential information responsibly both on and off-site. In some well-publicized cases, unsafe storage of confidential information on a subcontractor's desktop, on a former government official's laptop, and on a security auditor's laptop all provided the means to expose sensitive data outside the confines of a well-controlled IT environment. Needless to say, this also resulted in numerous and sometimes costly compliance violations and subsequent remedial action.

 Two publicized incidents of data theft involving medical information and Social Security numbers can be found at <http://www.federaltimes.com/index.php?S=2374521> and <http://www.federaltimes.com/index.php?2331714>.

Relation to SOX, HIPAA, PCI, and Other Compliance Regimes

The Sarbanes-Oxley Act of 2002, henceforth referred to as SOX, is formally known as the Public Company Accounting Reform and Investor Protection Act. This United States federal securities law was enacted to curtail a growing problem with corporate and accounting scandals, particularly with regard to how such organizations publicly disclose their internal accounting practices and how they document reporting controls. This far-ranging legislation establishes standards for all U.S. public company boards, management, and accounting firms and requires that the Securities and Exchange Commission (SEC) implement rulings on requirements for compliance. It also establishes the Public Company Accounting Oversight Board, an auditory body that oversees, regulates, inspects, and disciplines accounting firms that provide services to public companies.

As an IT professional, especially as such work involves storage, transmission, or access to sensitive information, you must comply with all governing rules and legislation, particularly because most financial reporting processes rely heavily on IT infrastructure to implement auditing and control mechanisms. Chief Executive Officers (CEOs) and Chief Financial Officers (CFOs) are responsible for corporate financial reporting, while CIOs are directly responsible for the security, accuracy, and reliability of information systems that store financial and other sensitive data or records. Any mismanagement of stored information either by leaking such data to unauthorized sources or exposing such data to unauthorized alteration, access, loss, or harm, is considered a violation of privacy and confidentiality.

Under HIPAA, there are provisions that also address the security and confidentiality of stored PHI. From medical records to payment histories, a covered entity is responsible for securing confidential patient information using a trio of safeguards under the security rule: administrative, physical, and technical.

Administrative safeguards ensure that HIPAA-compliant entities must adopt a set of information and security privacy procedures, which must reference management oversight in compliance with documented security controls and define authorized employees responsible for handling confidential data. Physical safeguards offer protection against onsite intrusions and specify control and monitoring mechanisms, including both hardware and software components, and define restriction controls for access to such resources. Technical safeguards ensure that information systems are protected in digital and electronic form, from the formats and protocols used to store and transmit data to the standards, procedures, and components used to secure these transactions based upon defined risks.

On March 16th of 2006, the Department of Health and Human Services made the Enforcement Rule effective. This rule assesses civil monetary penalties against parties that violate HIPAA rules and establishes procedures for investigations and hearings for violations.

The Payment Card Industries (PCI) Standards Council governs the development, enhancement, dissemination, and assistance with security implementation as it applies to payment account security. For organizations that process credit card payments, PCI promotes widespread adoption within various international industries, providing both tools and documentation to encourage strong compliance with the standards set forth by this council. Any company that processes, stores, or transmits credit card information is required to adhere to PCI Data Security Standards (DSS). Organizations that are affected include merchants, merchant banks, clearinghouses, and financial service providers.

A forerunner of PCI is the Cardholder Information Security Program (CISP) mandated by Visa in June 2001. CISP aims to protect credit card information as it is stored and ensures that members, merchants, and service providers follow information security best practices and comply with all governing standards. A collaborative effort by MasterCard and Visa earned CISP incorporation into PCI DSS and is accepted internationally by all major credit card issuers. Other prime examples include MasterCard's own Site Data Protection (SDP) program and Discover Information Security and Compliance (DISC), also derived from the PCI DSS.

Consequences for non-compliance go from mild to severe, and range from a \$500,000 per incident fine for theft of confidential data with additional costs and even jail time for corporate officers responsible for violations. Punitive damages and loss of company reputation are incidental to these violations.

Summary

File transfers may originate at any point on a network, from a client or a server node, at any time of day in either predictable or unpredictable fashion. Securing sensitive information begins at the source—right where the data is kept, processed, and transacted upon by client applications or any of a number of standard administrative or maintenance activities.

Observation of network activity through a network protocol analyzer aids in detection and identification of file transfer applications and protocols. An analyzer with high-level protocol dissectors is well-suited for this task, provided it has components that can identify all relevant traffic on the network segment under observation. Likewise, sophisticated network monitoring and management tools can also report on application and protocol activities, including:

- **Monitoring client traffic**—Simple observation of client traffic will reveal file transfer protocols and applications in use on the network. Typical host and port pairs for well-known service provider networks and file sharing protocols make a good initial start. High-level protocol dissectors can further assist in deciphering unknown port pairs or unusual source/destination port generation for signs of file sharing transactions.
- **Performing a protocol/port inventory**—Knowing a baseline of behavior helps to better identify unusual traffic patterns. Take inventory of known good port and service pairings and trace suspect traffic back to the source.
- **Who, what, when, where, and how**—Knowing who is using what file sharing method, what is being shared and when, where it is being shared, and how it is stored are important questions with need-to-know answers. Even a seemingly minor though careless act of retaining confidential information on a laptop or external storage drive unprotected and unsecured outside company property is an errant activity that has exposed untold millions of customer and patient records to unauthorized access.

In the following chapters, safe alternatives to standard FTP are examined further and options for securing similar though not as easily replaceable technologies are better explored. You'll also learn how to document the kinds of file transfers occurring on your networks and assess and mitigate threats and lessen vulnerabilities for those most likely to pose security or compliance risks.

Chapter 2: Planning Secure File Transfer Deployment

The first chapter set the background and laid the foundation for concepts explained and explored in this chapter. As you now know, today's typical enterprise file transfer environment is a result of long-term development and deployment of multiple forms of data transfer, often involving different hardware and software platforms. Typically, ad hoc solutions focus around the lowest-common-denominator or principle of least effort, which means leveraging native file transfer utilities to facilitate data transfers. Many times these tasks incorporate scripted interaction and scheduled automation to create periodic transfers for the most common and repetitive forms of shared or archived data.

FTP's popularity stems from its seamless platform integration, lack of interoperability issues, and nearly ubiquitous presence on everything from Windows workstations to IBM mainframes. Although this sort of "anytime, anywhere" availability seemingly makes FTP well suited to the task of file transfer, the FTP protocol and applications leave much to be desired in terms of centralized management, user accountability, data security, and real-time event auditing. Occasionally, this sort of convenience pays off in the near-term only to produce long-term security consequences by which confidential data can be revealed, intercepted, and possibly even tampered with by intermediate or external parties. There is also the possibility that an organization's FTP servers can be subverted for use as third-party storage repositories for illicit content or controlled as proxies for external network reconnaissance techniques. Eavesdropping, snooping, and hijacking attacks are three primary vulnerabilities that face improperly secured FTP installations.

Security Breaches

Security breaches happen all the time. Some instances are quietly dealt with as they occur, whereas others are too high profile or involve high-visibility targets (such as major corporations, organizations, or individuals) to avoid publicity. Some such situations now occur as a result of mandated reporting when security or customer confidentiality is breached, as the following examples reveal.

Mistaken postings of data occasionally occur, resulting in the disclosure of what should be confidential information. In January 2007, for example, an announcement from the Ohio Board of Nursing indicated that names and Social Security numbers for more than 3000 newly licensed nurses had been posted twice on the agency's Web site because of errors in sanitizing a posted report. Although other Ohio state agencies have elected to pay for a year of credit monitoring for those with compromised information, the Board of Nursing declined to do so.

The TJX Companies disclosed on January 17, 2007, that it had discovered the theft of credit and debit card information from its systems, which could affect customers of stores such as TJ Maxx, Marshalls, and others for whom the company processes card-based purchases. The intrusion resulted from a compromise of the companies' networks that handle credit card, debit card, check, and merchandise return transactions.


The University of Texas at Dallas discovered a computer attack on January 19, 2007, that may have resulted in exposure of Social Security numbers and other personal information for as many as 35,000 current and former students as well as faculty and staff members at the institution. This included 29,000 card holders at the University library, and a mixed group of 6000 other students, faculty, and staff. Though they have declined to discuss the details of the penetration, the university has indicated that it has taken steps to strengthen system and network security to prevent further attacks in the future.

These breaches do not account for other sources of exposure in previous years, such as malicious system cracking or poorly configured systems. For an excellent source of information about security breaches in general, please consult the archives of the Privacy Rights Clearinghouse at <http://www.privacyrights.org/ar/ChronDataBreaches.htm>.

Proper planning, implementation, and execution of secure file transfer methods, especially where compliance considerations are concerned, requires a multi-layered approach and thorough attention to detail. This chapter discusses several crucial points to help IT administrators address the issues involved in implementing secure file transfers within the network infrastructure.

Implementing FTP Inventory

It's not difficult to inventory plain-vanilla FTP transfers simply and easily by using a network protocol analyzer, but this activity may be better facilitated by way of a more robust traffic monitoring application such as Cisco Systems' NetFlow or the Multi-Router Traffic Grapher (MRTG). NetFlow is an open, but still proprietary, network protocol developed by Cisco Systems to operate Cisco IOS-enabled equipment for collecting IP traffic information.

 IP Flow Information Export (IPFIX) is described in Request for Comment (RFC) 3917 and other documents under development under the aegis of the Internet Engineering Task Force (IETF) IPFIX Working Group, as documented at <http://tools.ietf.org/wg/ipfix/>. IPFIX promises to replace NetFlow as the premier source for network traffic monitoring, analysis, and reporting, and is attracting interest from vendors other than Cisco.


MRTG is a freeware/sponsorware application that uses the Simple Network Management Protocol (SNMP) to collect statistical IP traffic information for accounting purposes. Both approaches have their unique strengths and merits and can report on the same type of information, which is useful when it comes to characterizing traffic and identifying traffic trends on any given network segment.

Several FTP-related or specific tools are also available. These include the AWStats log file analysis tool that is a member of the SourceForge family of projects (<http://awstats.sourceforge.net>); likewise, other free log analysis tools with strong FTP capabilities include Analog (<http://www.analog.cx>) and Webalizer (<http://www.mrunix.net/webalizer>). All these tools can report on who has been accessing an FTP server, for how long, and which files have been accessed, uploaded, or downloaded. Software Assist also offers free FTP analysis tools that include FTP Audit, which discovers all FTP servers in an enterprise or organization to help identify potential exposures, as well as FTP Sampler, a tool that may be used to dig into file transfers, identify sources, and document activity, including failed transfers and potential performance issues (<http://www.softwareassist.net>).

Taking one or more of these approaches will help administrators identify and evaluate file transfer activity on their networks, and provide a starting point for investigating sources of such activity. NetFlow can record a wealth of information about the network activities it tracks and report highly descriptive details about them, including statistical analyses, source and destination ports and addresses, protocols and flags, service values, duration periods, and more. FTP-specific tools, however, will provide the details that administrators need to identify and handle obvious, non-covert use of file transfer tools and utilities. Together, these tools combine to provide a complete picture of the situation.

As Chapter 1 stated, FTP may be characterized by a few predictable parameters through which all traffic can be detected, identified, and analyzed. The ease or difficulty in recognizing and identifying this traffic depends on placement of monitoring equipment and application behavior on the network under observation. Not all file transactions are FTP-based, however. Some transactions do not even employ a client-server model, opting instead for peer-to-peer arrangements whereby endpoints operate in an ad hoc fashion, thereby merging client and server roles and switching back and forth between them. In fact, some peer-to-peer services are structured to switch randomly among multiple sources for data during file transfers to purposefully mask that very activity!

It is worth recalling that TCP ports can be either permanent or short-lived. Servers listening specifically for FTP traffic may be permanent in the sense that a port is always kept open, awaiting inbound connections from calling client applications. That said, active and passive transfer modes directly influence the nature of any particular FTP exchange.

 Refer back to Chapter 1 for an illustrated explanation of these modes of operation.

Client-side ports remain open only for the duration of a transfer and are not bound to a specific “listener port” as are server-side applications. However, server-side ports may also be temporarily allocated to some higher-numbered port, following the long-standing UNIX tradition of using low-numbered ports (below 1023) for privileged processes only. Depending on the transfer method employed, an FTP server may also make a temporary port allocation for the duration of an exchange, which serves only to complicate spot-check analysis. For this and other reasons, real-time traffic monitoring solutions are described and discussed later in this chapter.

On-site network appliances are ideal for observing localized traffic usage patterns where file transfers occur, but this is a naive perspective at best. Large-scale site-to-site file transfers may encompass many endpoints across multiple disjoint regions, which may be controlled by an off-site management application that negotiates cross-site server-to-server exchanges. Several combinations of software and hardware must often be employed to identify network-driven file transactions.

For example, Javvin’s Packet Analyzer is a Java-based application that can track and report on four specific types of traffic, two primarily interesting types being FTP and HTTP. The Packet Analyzer permits an administrator to view reconstructed content and offers a simple tabbed interface and free trial-use period to evaluate its capabilities. Wireshark (previously known as Ethereal) is a free full-fledged Ethernet network protocol analyzer with a variety of filtering, sorting, and dissection capabilities as well as advanced analysis features. The former is a commercial product written in a cross-platform language, the latter is a free but fully capable cross-platform solution. Both provide the ability to observe and monitor network traffic. There are many similar solutions available on the open market, each with its own options, benefits, and drawbacks. What remains consistent across all of them is the training necessary to make best use of whatever products and technologies are acquired to capture and categorize network traffic by site or location.

Capture and Characterize Network Traffic by Location

Knowledge about where organizational units and their employees are housed is essential when determining the role and purpose of any given network segment. In other words, it's important to know where network segments are located; which of them have desktops, servers, and other devices attached to them; what they use them for; how those segments are attached into the network core; how they obtain Internet or wide-area network (WAN) access; and so forth.

For example, a financial institution may organize its accounting department assets into logically and physically separate office spaces and network segments, perhaps tied together through a local switch, itself attached to a corporate backbone. Web-based consulting services may also be separated internally and externally either as standalone internal servers and services, or perhaps operating in a demilitarized zone situated between the internal firewalls that guard the corporate network, and the external firewalls or routers that attach to one or more wide-area links. Every department also typically has its own separate physical and logical access to different kinds of confidential client data. Therefore, each department must operate under the same canopy of compliance but with slightly different procedures.

Traffic captures should be characterized according to their points of origin. You must give special attention to how and where confidential information passes through each department. Web-based transactions may seem transient and sporadic, while internal database accesses often appear to be more deterministic and predictable. Not only should ongoing traffic monitoring be your guide to finding file transfer activities and users, local system accounting and log files also leave an electronic trail that you must examine to establish usage types and trends for each given server.

On BSD, Linux, UNIX, and Windows hosts, an application such as Sawmill can process log files and generate dynamic statistics reports. Sawmill can parse UNIX FTP logs, import them into Symbolic Query Language (SQL) format in bulk, and report on a variety of details including date and time of a given transaction, file type, host name, domain name, geographic location, authenticated user, transfer mode, and the direction for a given data exchange (or file transfer). This kind of information arms network administrators with exactly what is needed to evaluate file transfers as they occur locally on any machine under their purview. Other FTP-specific log file analysis tools such as AWStats, Webalizer, and FTP Sampler might also be brought to bear, as described earlier in this chapter. The Institute of Internal Auditors (IIA) has also published a very informative article entitled "11 Steps to an Effective FTP Audit" (<http://www.theiia.org/ITAudit/index.cfm?catid=21&iid=513>) that provides an overview of this entire process and includes a list of default FTP server log files that administrators could (and should) search for along the way. This kind of information can be quite illuminating, particularly when used in tandem with a suitable log analysis tool.

Perform a Port and Protocol Inventory

A port and protocol inventory is best handled by network traffic graphing and monitoring utilities placed at major junctions of the network, such as enabling NetFlow on WAN-attached routers or at key infrastructure cross points, or by using MRTG to interrogate SNMP-capable network devices. Host-based protocol analyzers such as the robust open source Wireshark application can also perform spot checks, providing additional traffic flow detail and analysis reports for file transfer-related protocols and ports in use on any given segment. Wireshark also offers numerous relevant protocol dissector plug-ins that can assist network enumeration procedures and traffic identification, characterization, and security checks.

The objective is to determine what ports are in use on the network segment you're characterizing, then to zero in on protocols or ports that can indicate file transfers may be occurring or involved in observed activity. The idea is to identify all potential file transfers, to determine their sources and destinations, and ultimately to identify the applications used to perform such transfers. A solid, comprehensive port and protocol tool is invaluable when it comes to tracking down this information.

Trace Items of Potential Interest

Large-scale one-to-one file transfers, such as explicit FTP transfers, are the most obvious manifestation that you're seeking throughout this exercise. But one-to-many mappings such as peer-to-peer transfers, which may involve a single peer client on one end but any number of peers on the other end, tend to follow non-linear and almost non-deterministic patterns. Such is the case with Torrent-based peer-to-peer transfers that can select portions of a single file from numerous active BitTorrent participants and effect transfers from all of them in no particular order. This explains why identifying certain types of file transfer can be more difficult than others. It also explains why you should note anything that stands out against a normal network traffic baseline, such as the sudden and unexpected appearance of BitTorrent metadata files (which typically end with a .torrent extension) and for which related software typically listens for download requests on port numbers in the range from 6881 through 6889 then work through each such annotated transaction manually. This also helps to affirm the contention that identifying applications in use, or associated with specific ports and socket addresses, is also quite important.

Track Unknown Items Back to Their Original Locations

Be aware of any unidentified sources of traffic. Not every protocol is well known, well behaved, or sufficiently well established on the network to make enumeration easy. There may be some proprietary protocol active on your network that lacks a proper protocol decode module for your network analyzer of choice, or that uses non-standard ports for communications. Where a packet trace yields little evidence or information to describe the kinds of network activity that are underway, protocol analysis based upon specific circumstances takes on a more personal and perhaps more intrusive and hands-on feel. You must trace difficult transactions back to their points of origin and isolate one or more involved endpoints, then zero in on all active services in each system's process space. When all attempts at identification and characterization fail—and they sometimes will, though not often—you can always block the protocols or hosts involved and wait to see what kinds of complaints may be forthcoming. Obscurity of this nature is seldom associated with mission-critical applications, but it's always wise to check first with in-house developers just to make sure what you're observing isn't simply evidence of a poorly documented internal application before shutting down such traffic, even temporarily.

Identify All Types of File Transfer

For the most part, inventorying most file transfer protocols and applications is pretty straightforward. (To help make this process easier, this guide provides an appendix that itemizes all well-known file transfer and peer-to-peer protocols and related ports.) Many forms of file transfer contain obvious traffic signatures that may be readily identified by protocol analyzers and protocol analysts, intrusion detection systems (IDSs), firewalls and traffic monitoring appliances, and so forth. Ultimately, ports, protocols, and payloads are the three key elements by which file transactions may best be recognized and identified.

That said, not all forms of file transfer are neatly parameterized and clearly defined on the network. One easy way to evade simple detection by a network protocol analyzer is for client and server applications to choose non-standard ports and then to encrypt end-to-end communications. Services may run only for short periods of time, awaiting the arrival of some predetermined sequence of packets (called port knocking) or some other signifying event before negotiating any active connections. But such behavior is often indicative of malign forces at work and should trigger a full-fledged security investigation once they're detected. Most routine, normal file transfers will be entirely overt, and much easier to recognize and identify.

Many regulations require that data shuttled across public networks be analyzed for risk of exposure to PII and that any at-risk data be encrypted to meet security standards to avoid its exposure to unauthorized third parties. In some cases, this may involve encryption of data while it's at rest inside a file system somewhere and again whenever it is in transit from an authorized sender to an authorized recipient.

Perform a Platform Inventory

Proper documentation is part of the compliance process required by many regulations. This task includes describing and detailing all the security controls, policies, procedures, and facilities used to ensure compliance. Documentation for organizational hardware and software assets is crucial to understanding your security landscape. It's essential that you consult such documentation as part of your research process, and that you be prepared to amend and update such documentation as you conduct your own research (or at least to submit such information to the proper authorities).

Security Documentation


Documenting security information is crucial to standard IT management and should have been performed as a standard of due care already. Establishing a comprehensive, up-to-date inventory of hardware and software components related to security also assists in the detection, discovery, and development of your IT infrastructure and the activities that take place upon it. This means an inventory of devices and applications down to a highly granular level of detail including the directories, files, permissions, registries, databases, and computers involved, along with all relevant configuration parameters. Know what hardware and software platforms comprise the corporate environment and pay particular attention to software versions. Each operating system (OS) has its own methods of implementing security controls and may be subject to weaknesses or exposures depending on patch, hotfix, or update levels in place. Although the concept of security and compliance is common to all platforms, the components used to implement or verify compliance will typically vary on a per-platform basis (if not from situation to situation, as is very often the case).

Compliance Management

A compliance management solution such as Cendura's Cohesion software utilizes the Control Objectives for Information and related Technology (COBIT) framework and assists IT staff and auditors in the design and review of IT process controls supporting IT governance and regulatory auditing initiatives. Cendura provides continuous support for planning and organizing IT environments, acquiring and implementing program development, delivering and supporting application operations and data access, and monitoring and evaluating these environments. Their software solution, Cohesion, can inventory and audit application infrastructures, compliance management automation, policy compliance enforcement, and many other relevant aspects of regulatory compliance controls.

Identify Platforms and OSs in Use

UNIX and Windows are among the more common platforms deployed in anywhere from small business to large-scale enterprise network environments. In addition, mainframe platforms, typically the IBM z/OS systems, play an important role in the enterprise data transfers and storage. The following list highlights several names for the many OSs found in modern network environments. This list is not meant to be comprehensive and encompasses only more typical OSs found on most networks.


 What is interesting to note is that most of these platforms include built-in FTP and other file transfer clients, and nearly all of them support similar server-based applications as well. Thus, the means and the opportunity for file transfers are invariably present, which also means they can sometimes show up in interesting and unexpected contexts.

There are many variations of each UNIX and Windows-based platform, many of which are still in use today. Some of these operating systems include:

- IBM z/OS hosts and environments
- UNIX and Linux—FreeBSD, NetBSD, and OpenBSD; Debian, Redhat, SuSE Linux (among others); Mac OS X; AIX, HP-UX, SCO, Solaris, System V; OS/2 Warp
- Windows—9x, NT, 2000/2003, XP, and (more recently) Vista hosts
- MacOS—There is the original, proprietary MacOS in any of a wide number of versions (1.x through 9.x)

The least intrusive method for inventorying these platforms involves consulting administrative documentation that enumerates and describes their installations (where applicable and available, as will be the case in environments in which such documentation is both present and current). In the absence of complete, current, and accurate documentation, you must turn to either passive or active means for fingerprinting endpoints and performing platform discovery that require observation of network traffic (passive discovery) or some sort of stimulus-response interrogation (active discovery).

Two open source tools used for large-scale fingerprinting of network endpoints include the passive OS fingerprint application (p0f) and network mapper (nmap), both very popular in their respective camps.

 For more information about the passive OS fingerprint method, p0f, visit <http://lcamtuf.coredump.cx/p0f.shtml>. For information about nmap operation and usage, see <http://insecure.org/nmap/download.html>.

With p0f, a host is discovered during initial packet synchronization (SYN) and identified by default start-up values contained in these packets. p0f can determine the distance to a remote host and the structure of a local or foreign network.


In contrast, nmap seeks to interrogate a remote host actively and performs network discovery differently. That is, nmap uses a battery of tests to provoke the remote network stack into revealing specific properties that are more-or-less unique to each platform. Thus, nmap can typically determine a multitude of things right down to a particular version and patch level for a given OS. The nmap application is fully capable of fine-tuned probes that can distinguish among OS versions and discriminate between workstations and servers as well as routers and printers, among various other network appliances and intermediate devices.

Identify Secure File Transfer Solutions

There are many types and variations of secure file transfer solutions available on the market, each with its own advantages and disadvantages that must be weighed and considered carefully on a case-by-case basis. What is applicable for one organization may be deemed unsuitable for another. In fact, there is no one-size-fits-all solution.

A variety of secure transport implementations, for example, can provide consolidated and centrally managed systems for the management of secure site-to-site and application-to-application file transfer activity. Such exchanges occur over FTP, SFTP, HTTP, HTTPS with Transport Layer Security (TLS) or Secure Socket Layer (SSL), secure shell (SSH) via SFTP and Secure Copy (SCP), and the Applicability Statement 2 (AS2) specification. They can operate across a variety of platforms, are compatible with multi-tiered networks, comply with both SSL and Lightweight Directory Access Protocol (LDAP), operate with Active Directory (AD) file routing and handling, and integrate with Java 2 Enterprise Edition (J2EE) applications. Perhaps most important, such secure transport implementations also typically conform to industry, corporate, and government regulations including the Sarbanes-Oxley Act (SOX), the Gramm-Leach-Bliley Act (GLBA), and the Health Insurance Portability and Accountability Act (HIPAA).

AS2 is a specification that describes the safe, secure, and reliable transport of data via the Internet as defined in RFC 4130. This specification describes how to transport data safely and reliably across the Internet, where data consists of electronic data interchange (EDI) messages or any other message type. AS2 envelopes a message using digital certificates and encryption methods for transport between client and server application software to safely push confidential data across the network or Internet. AS3 is a current working standard that combines AS2 capabilities with FTP and Secure Multi-purpose Internet Mail Exchanges (S/MIME) to establish secure real-time connections between business partners using SSL.

 AS2 RFC 4130 may be perused by visiting <http://www.ietf.org/rfc/rfc4130.txt>.

Disadvantages to an AS2 approach are that endpoints rely on static IP addresses and fixed locations, imply specific firewall considerations, and require equally specific administrative expertise. These drawbacks automatically rule out the roving nature of mobile computing platforms and infrastructures. Endpoints cannot pull data or continue failed or truncated file transfers and require certificate management to establish secure connections. The costs of AS2 software can be high but may offer low return on investment (ROI) for high transaction volumes. Also, AS2 only works across native TCP/IP networks; thus, older X.25 and ISDN-based technologies can't use its services. Nevertheless, plenty of high-profile companies such as Wal-Mart have opted for AS2-based technologies to replace their former reliance on dial-up technologies for purchase orders that occur at periodic intervals to avoid the otherwise costly implementation of a value-added network (VAN).

The Odette FTP (OFTP) specification offers an alternative to AS2. OFTP 2.0 describes secure transfer of business documents via the Internet and includes X.25 and ISDN networks. The details of OFTP v1.3 can be found in RFC 2204, while v2.0 is currently available as an Internet draft. OFTP works either point to point or indirectly via a VAN, where a single OFTP entity can place and receive data, exchange files in either direction, and operate in push or pull mode, unlike AS2. Furthermore, OFTP can encrypt and digitally sign message data, request signed receipts, offer high levels of file compression, and, when run over TCP/IP networks, make delivery using TLS. OFTP's advantage for business applications stems from its origin by and for the automotive industry for e-Business communications and engineering data as a protocol for automated EDI. Although it was initially designed primarily for the automotive industry, OFTP can also be found in retail, petrochemical, and tax submissions and banking sectors, among others. OFTP 2.0 provides session and file security and secure authentication and offers more robust capabilities than its AS2 counterpart.

 OFTP RFC 2204 resides at <http://www.ietf.org/rfc/rfc2204.txt>. To read the latest Internet draft, dated September 2006, visit <http://www.ietf.org/internet-drafts/draft-friend-oftp2-03.txt>.

Here again, interoperability is a key ingredient. Interoperability within the network infrastructure is as crucial to business operations as site-to-site interoperability across multiple network infrastructures. The protocols, specifications, standards, and hardware must share one or several common denominators to pull everything together. Such a process is much more involved with spoke-hub scenarios such as health care payer-provider derived systems versus more collaborative peer-to-peer platforms.

Look for Intersection Across Platforms

Intersection across platforms refers to common components or capabilities that are not unique to one platform but ordinary to many. One such major intersection is in the TCP/IP network stack utilized by many networked computer systems from BSD to Windows on notebooks and laptops, workstations and servers, as well as network appliances and other widely used devices. Basic connectivity between any two machines can be created where TCP/IP protocols and equipment provide a common denominator. That said, interoperability issues are likely to arise in the following key areas:

- Authentication methods
- Communications protocols
- Software applications
- Data and information

Communications Protocols

Systems should be able to pass and receive information in some generic manner without any pre-programming of sender or recipient that is party to such transactions. The parameters for both originator and destination endpoints need not be explicitly known by both parties to create ordinary transactions, just as a Windows workstation client should not particularly care that a UNIX server acts as its host platform. Both platforms are fluent in TCP/IP as a primary bi-directional communication scheme and quite capable of sharing files in a number of ways, most notably FTP.

Where EDI transactions are concerned, such as in the health care industry, this may require an advance arrangement that employs one or more designated communications protocols to permit out-of-the-box interoperability between source and destination. This sort of interoperability is not covered in HIPAA and indeed there is a significant gap in the health care industry regarding an all-purpose protocol framework within which to achieve such standard communication. A direct side effect of this set of circumstances is that each trading partner (a covered entity—or CE, in HIPAA terms, meaning a pharmacy, any health care provider, and any insurer that handles medical claims) necessitates an inefficient validation process for each and every implementation as new partners come online.

Software Interoperability

Software interoperability ensures that two or more products are tested for compatibility and provide an interoperable user interface and interoperable commands; likewise for scripts, macros, and data files. In our imperfect world, we cannot always have access to development procedures, standards, and specifications, nor will the most extensive testing always make all interoperability issues apparent.

Here again, FTP's success owes to its seamless interoperability across different hardware and software platforms. This popularity is heavily battle tested and has evolved from years of development and deployment in large-scale university and enterprise network environments, though as we've already observed repeatedly, FTP is also not without its drawbacks and limitations. But this same desirable all-purpose characteristic is essential for any secure solution destined to fill the role of an enterprise-grade replacement file transfer solution.

The secure transport approach described earlier in this chapter works across disparate systems using platform-independent open standard protocol specifications, as do SSH-based approaches. These replacements simultaneously eliminate vendor-specific tie-ins to any architecture, infrastructure or framework and extend security and regulatory compliance coverage to virtually all current platforms in a business or enterprise environment. Financial transactions, critical business files, large documents, and EDI transactions can then take place securely over the Internet and across private IP networks. This setup permits organizations to supplant expensive legacy methods of data transfer and to avoid the potentially costly establishment of virtual private networking (VPN) technologies, leased-line subscriptions, or VAN deployment—and particularly avoid unsecured FTP transactions.

Data and Information

Last but not least, you must also consider the importance of data and information interoperability. Particularly where different platforms, interfaces, and applications are used, generic or portably defined documents and information specifications provide a basic framework for interoperability. At the data level, where data produced by one product is readily understood by a product designed for an entirely different platform, both products can share and process common information. Information passed within and about this data can also be produced and presented in platform-independent ways.

Email is a common point for information exchange, where interoperable clients exist for virtually every kind of computing platform. An email composed on a Windows-based Outlook client can be formatted, delivered by a UNIX-based SMTP server, then read on a Linux-based Thunderbird recipient. Defining appropriate data sets, assessing the correct level of data representation, and evaluating the right solution are all part of the same interoperability examination process when it comes to file transfer.

Making a Technology Selection: Performance Issues

Several high-profile data theft cases in the recent past sent an industry-wide rude awakening that underscores the need for heightened awareness and better security procedures for storing and transmitting confidential information. Perimeter security is not enough; when data passes beyond the scope of perimeter-based security mechanisms, it can be exposed to tampering, interception, and possibly even modification or manipulation. Tampering can happen at any time—an unattended notebook or external drive stolen off-site, an unsecured connection on a foreign network, or even a momentary physical access by an unscrupulous third party all provide opportunities for tampering. Not to mention bad things happening, by accident or maliciously, by authorized network users inside the perimeter.

These examples may seem naive at first blush, but consider a traveling IT professional who will pass through numerous airports and cabs or drive multiple rental cars. At any point during this sequence of activities, his or her notebook could be left unattended, only to be stolen by some opportunistic criminal. A publicly accessible wireless hotspot or Internet cafe may make a perfect temporary arrangement to access the Internet and upload or download information, but may also provide eavesdropping opportunities or involve connections through rogue or attacker-modified access points. And if his or her notebook should break on the road, there is always the possibility that the notebook will end up in a third-party technical service department, where unauthorized outsiders could dig into its contents.

It may be unrealistic to audit where and when all pieces of confidential information are left exposed, even with the strictest auditing regimes and controls in place, so it may never be fully known what confidential data remains unsecured on a notebook's drives at any given time. The one real certainty is that only data stored or transmitted in a cryptographically secure format can provide adequate protection against leakage to or access by unauthorized parties.

When it comes to implementing cryptographic solutions within a business environment, one of the first queries is how they address business needs. First and foremost, an encryption solution will resolve most rigorous compliance issues with regard to data storage and collection. Ensuring privacy for credit card transactions is exemplified in Requirement 3.4 of the PCI Data Security Standards (DSS), which requires that sensitive cardholder data be rendered unreadable anywhere it is stored. That said, many regulations also provide a “safe harbor” clause for companies that implement encryption. When a breach of security occurs, companies acting in good faith will have limited or no liability for such incidents.

Performance arises as another issue for encryption deployment, especially where large-scale enterprises or multiple-client organizations are concerned. This issue is discussed further in the following section.

Costs of Encryption and Compression

The performance of a given cryptosystem or encryption solution will be directly related to the implementation approach. There are host and application-based encryption solutions that burden local general-purpose processing resources during normal operation. A CPU already under load from other tasks will be greatly impacted by localized software-driven encryption methods that rely on its processing capability, unlike an appliance or storage-based solution that exports this burden to specialized standalone hardware resources. The first solution will affect notebook, desktop, or workstation computers at best; at worst, software-based encryption often requires significant additions or major overhauls to existing server hardware.

In general, the following types of encryption solutions are available to plug existing gaps in data security and to help meet compliance requirements for data privacy and confidentiality and customer security:

- Host and application-based solutions are implemented on each at-risk computer that requires cryptographic components for secure operation. This method introduces the least amount of change to an existing hardware infrastructure at the lowest possible performance level. For underpowered systems, encryption and compression routines may be too resource intensive and require substantial hardware upgrades to bring such systems up to compliance standards.
- Storage-based encryption solutions utilize native processing resources to apply encryption and compression routines. As such, this solution leverages much of the operational overhead away from hosts and host applications to achieve the same or similar goals. This method introduces a moderate amount of change to existing hardware infrastructures and may require additions to or replacements of client and server software.
- Appliance-based encryption solutions are implemented in specialized standalone units utilizing hardware-accelerated encryption and sometimes compression routines. This method also introduces marginal change to an existing network infrastructure and minimal disruption to services. Some units are capable of interfacing different backend storage technologies including NAS, FC, and IP SAN. As you would expect from the best-performing solution, it is also the most expensive to implement.

Budget and performance costs are largely determined by the implementation approach taken for each solution. Host or application-based solutions may include per-seat or per-user licensing costs and additional periodic upgrade fees, where turnkey storage and appliance solutions involve upfront product costs and may also entail subsequent support and service subscription fees. Hardware-accelerated encryption solutions are considerably more expensive than software applications that use general purpose computing resources. Furthermore, each such solution brings with it a unique set of challenges and potential changes to the existing hardware or network infrastructures, all of which must be factored into the selection process.

Storage and appliance solutions continue to grow in popularity as an alternative to resource-consuming host or application implementations. Some offerings are built around gigabit-capable encryption appliances that integrate transparently into NAS, SAN, DAS, and tape backup environments and provide cryptographically secured storage compartmentalization. Other similar solutions use enterprise-class network appliances that non-intrusively safeguard critical database information and that deliver enterprise-class transparent cryptographic transmission and storage. Each of these devices can pass data compressed and encrypted at line speeds for superior performance with minimal impact to other networking resources and tasks. There are also specialized parts such as encrypted tape storage solutions and encrypted notebook drives that provide coverage for data at rest that may venture off-site beyond perimeter security controls.

Identifying and Minimizing Potential Bottlenecks

Encryption merely complements or expands an existing IT security posture and should be applied anywhere sensitive data is stored, collected, and maintained. This protection comes at a considerable cost: good software solutions can impose additional overhead in terms of resource consumption and levy performance penalties that range from mild (barely noticeable) to 15 to 20 percent reductions in processing and throughput, and effective hardware solutions to accelerate good software solutions are costly to purchase. That said, not all SSH implementations impose noticeable performance reductions, and most enterprises and organizations are happy to accept minor performance degradation in exchange for improved security and the ability to hit mandated compliance targets.

Performance suffers wherever compression and encryption tasks double-up alongside other process-intensive tasks. Workstations with general purpose CPUs that process application data with these other two resource-consuming tasks can become overburdened. Network transactions also lean on the CPU unless some kind of onboard hardware acceleration is built into the network interface, which is more the exception than the rule. But indeed, both of these chores can be offloaded to add-in PCI cards for those willing to purchase and deploy them. Such specially designed cards can deliver moderate-cost hardware compression and encryption solutions with throughput suitable for use on T3, E3, and Fast Ethernet technologies.

When you examine your situation and start working toward a solution, it's important to assess what data realistically needs cryptographic coverage. You should perform a risk analysis and restrict the scope of cryptographic coverage to only the most at-risk business assets. Not only is this a good practice, it is also required by numerous regulations. Not every file and network transaction must be cryptographically secure; neither must every network endpoint. Establishing an appropriate level of precaution against illegal tampering or interception of safely stored data reduces liability, remains within compliance requirements, and reduces the waste and potential overuse of resource-intensive encryption and compression products.

Benefits of Hardware Acceleration

Storage security appliances can be placed inline between storage volumes and client endpoints to reduce processing power required for local host-based resources. Hardware acceleration and integrated encryption components built-in to these appliances facilitate the compression and encryption routines transparently for network-driven transactions. They can also provide excellent performance returns for the investment involved. In many cases, such appliances introduce little or no changes to existing network infrastructures, provide file access monitoring, and might possibly even restrict unauthorized use. All of these factors add up to sizable potential gains. Storage appliances, network appliances, and add-in peripherals all tend to incorporate some form of hardware acceleration to drive compression and encryption routines while also delivering reasonable levels of performance to enterprise users.

Making a Technology Selection: Security

Trust is an important and considerable factor in any cryptographic exchange. We trust that the cryptographic algorithms used in an exchange are reasonably strong, that we have chosen appropriately strong passwords, that the implementation is relatively invulnerable to most forms of attack, and that both parties involved can be identified, authenticated, and accounted for. Sometimes a trust relationship is formed in a small window of time between the originator and recipient of a message, as happens with online retail outlets where a customer renders electronic payments. It's next to impossible to verify anyone's identity one-on-one through an Internet connection alone. Therefore third-party entities are entrusted to bind and manage identities mapped to credentials so that authorized parties can establish confidentiality with one another and begin transmissions in a cryptographically secure manner.

Key Management Schemes

Standards and protocols such as IPsec and TLS/SSL use server-level Public Key Infrastructure (PKI), passwords, and shared secrets to provide a secure foundation for network communications. By design and by default, these solutions do not always match up to the comprehensive security, scalability, or manageability required by compliance regimes and provisions offered by true end-to-end PKI solutions.

To ensure that these point-to-point transactions across diverse platforms and applications are secure, users can use digitally signed transactions to provide the paper trail necessary to meet compliance standards. Provisions for public key encryption and digital signature services and manageability are made possible through the PKI. An organization establishes and maintains trustworthy environment network partnerships through the PKI in a transparent manner—that is, an end user is not required to understand how PKI works to take advantage of its services.

An alternative approach to public keyed information exchange is a web of trust scheme, a method of using self-signed certificates and third-party verifications for such certificates. Pretty Good Privacy (PGP), GNU Privacy Guard (GnuPG), and OpenPGP are increasingly popular as a result of their inclusion in secure email and Web transactions, among other practical uses. Simple PKI (SPKI) is born of three independent efforts to reduce the complexity of X.509 certificates and avoid the anarchy of PGP's web of trust. SPKI binds people and systems to their respective keys using a local trust model like the web of trust with integrated authorization capabilities but has no definition for the role of Certificate Authority (CA).

Public Key vs. Private Key Infrastructures

PKI encompasses a set of developed and formalized standards, products, solutions, policies, and practices for secure data transmissions over unsecured lines. As such, PKI is an arrangement that provides for trusted third-party examination, evaluation, and verification of user identities. This method of secure transaction binds public keys to user identities and exchanges certificates on behalf of properly authenticated and presumably legitimate parties. PKI is a stable cryptographic framework that sees widespread usage and deployment across the often unruly Internet and appears in several types of products from email and HTTP clients to VPN concentrators and wireless infrastructures.

To examine basic PKI features reveals many of the reasons that private key technologies do not work in the context of mutual public exchanges. For starters, how does one safely exchange private keys, ensuring that no party has intercepted or modified such keys? With a single all-purpose private key, anyone can use it to encrypt and decrypt messages keyed with its content. Therefore, how can you trust that all subsequent communications are made only by the intended recipient of the private key? These and many other questions are resolved by using some form of PKI, where each implementation has its own solutions for each problem associated with public key exchanges.

There are many ways to safely and securely move data; exchanging private keys is not one of them. Where a cryptosystem uses the Digital Encryption Standard (DES) algorithm to create all-purpose private keys, RSA uses an asymmetric algorithm to achieve the public/private key pairs necessary for PKI. Digital signatures and certificate systems bind the original party with the public encryption key together with some key distribution system with management facilities such as a CA or web of trust.

Benefits of Centralized Key Management

Key backup and key escrow are two features provided through centralized key management facilities. In key escrow systems, a third party can obtain decryption keys for encrypted data. Such need arises when a federal investigation requires access to information secured by encryption keys kept by an escrow service, or when an authorized user accidentally loses access to or destroys a key that must be replaced so that data encrypted with that key remains accessible.

Key backup solutions are ideal in less legally binding cases, such as when employees change departments or employers and for long-term data storage. Passwords and logon credentials change over time and may easily be lost or forgotten, unless little variation is introduced into your routine. Your password or passphrase of 10 years ago surely isn't the same one you use today. But your public and private keys are another matter altogether. Thus, key backup serves an important recovery purpose—to keep secured data accessible to authorized parties when the original party or original key is no longer available.

PKI and Interoperability Issues

Interoperability has proven to be a crucial issue for PKI implementations. It's simply not safe to assume that PKI components from vendor A will always work with those from vendor B, or that components from one or both A and B will also work with vendor C, and so forth. Unfortunately, this puts IT professionals in the interesting situation of having to seek definite proof of interoperability as part of the purchase process, or to stick to products from a single vendor. The only other sensible multi-vendor alternative is to purchase small numbers of products for test or pilot implementations, to weed out products or solutions that don't prove to interoperate properly, and to construct workable systems through repeated testing that may not exactly qualify as "trial and error" but often looks like something close to that by the time all the necessary work is done.

Another approach is to look for products built around common underlying PKI technology, which may not require a single-vendor architecture but ends up employing a single-supplier PKI nevertheless. This explains why some security companies make a business out of licensing certificate toolsets specifically to support the Internet X.509 PKI certificate and certificate revocation list as specified in RFC 3280, aka PKIX for Public Key Infrastructure (X.509). Although it may not always be possible to purchase PKI software or components from the same vendor, to meet client needs for email, VPN, file transfer, and so forth as well as to support PKI in an enterprise (CAs, revocation list management, Online Certificate Status Protocol services, and so forth), make sure that at least some of the different components employ common underlying PKI technology. Doing so simplifies the job of vetting interoperability and may even make it possible to demonstrate interoperability without extensive testing. The most responsible vendors maintain extensive libraries of compatibility or interoperability test documents that are well worth researching as you work your way into a practical and manageable security implementation, especially where PKI is involved.

Avoid Security Through Obscurity

Simply put, security through obscurity is wishful thinking and doesn't work. Here, obscurity means the act of hiding implementation-specific details governing the operation of a presumably secure protocol, application, or infrastructure. This might be a software product, a communications protocol, or a topological arrangement of network devices. By leaving crucial details invisible to observers, the idea is that little can be derived, discovered, or determined about whatever security measures are in place. The assumption is that nothing shows to reveal its secrets or to exploit its weaknesses. In actuality, this couldn't be further from the truth.

Secrecy as a method of security is flawed for many reasons. Though a system relying on such logic may have theoretical or actual vulnerabilities, its designer makes the assumption that its weak points are unlikely to be identified by attackers because no properties about such conditions have been revealed. However, this does not mean vulnerabilities cannot be found, and certainly does not mean such a system can safely be entrusted with protected information or allowed to operate independently within a secure setting.

There is nothing wrong with public or private scrutiny of a security or cryptographic system's source code. In fact, there are benefits to having professional developers, security experts, and systems analysts review a given application, protocol, or framework for weak points that could lead to vulnerability or exploitation. A statement to this effect, known as Kerckhoff's principle, was made in 1883 by Auguste Kerckhoff, "A cryptosystem should be secure even if everything about the system, except the key, is public knowledge."

Indeed, a cryptosystem is much like any other security system and should always be reviewed by many specialists and experts in various security fields to ensure proper fitness during operation. World-renowned security expert and cryptography specialist Bruce Schneier extends this reasoning to all security systems when he states the following, "Kerckhoff's principle applies beyond codes and ciphers to security systems in general: every secret creates a potential failure point. Secrecy, in other words, is a prime cause of brittleness—and therefore something likely to make a system prone to catastrophic collapse. Conversely, openness provides ductility." This notion of ductility, or structural rigidity and resistance to deformation, is vital to every security framework and infrastructure.

Therefore, whenever you select technology for enterprise deployment, insist on security-evaluated or certified software products. When researching a potential security product for use within a production environment, inquire about its internal security policies and practices and whether the system has been audited by any external agency or testing laboratory. In lieu of such assurances, you might have an independent auditing firm inspect or test the product for potential problems. Accept no vendor assurances to the contrary: such systems must be verified and validated. Any sign of resistance, short of non-disclosure agreements to third-party sources, should be viewed with suspicion. After all, the safety and livelihood of your confidential business practices are at stake and there is no sense in leaving such critical items of interest unchecked or untested.

Compliance and Security Audits

In terms of networking, compliance refers to the state or act of conforming with or agreeing to provide adequate security and privacy according to mandated and regulated specifications. For IT departments, the following three areas are most affected when processing confidential data:

- Data authorization and access
- Data retention
- Data retrieval

Network and host-based storage infrastructures are susceptible to damage or loss of data, potentially vulnerable to security breaches, and challenged by the dynamic nature of an ever-changing security landscape. To further complicate matters, organizations that deal in PII are perpetually regulated and routinely examined for compliance. IT's major role and responsibility in this process is to maintain a strong security posture that satisfies both sides of the at-risk equation. Execution requires a methodical approach involving risk assessment, implementation evaluation, and periodic monitoring practices. It also strongly urges the need for enhanced security awareness, training, and expertise, and demands the following IT issues:

- Uncovering existing problems
- Mapping potential solutions to business values
- Justifying ROI
- Targeting high-return countermeasures
- Rigorously using management-based measurement and monitoring methods

With SOX, GLBA, and HIPAA now part of the common business vernacular, a proliferation of many more mandatory regulations and regulatory regimes give rise to innumerable IT security requirements, some of which are vague and difficult to translate into technical controls. Regulatory compliance requirements tend to be broadly worded and use generalized concepts, while asset compliance policies are precise and accurately descriptive. Where a regulation is worded too broadly, there is the potential for misinterpretation of the definition of compliance depending on the subject making the interpretation. The evaluation of a solution that correctly addresses all such broadly defined mandates may not even improve the existing security stance. This is what makes internal and external audits so important to ensuring compliance—both give ample opportunities to identify ambiguities, resolve misinterpretations, and to ensure that implementations adhere to both the letter and the spirit of governing regulations. This also explains why compliance experts stress the need to treat compliance needs as an opportunity to improve overall IT operations, not just to bring security up to minimally acceptable levels.

Securing IT assets requires a specialist, someone who possesses a specialized knowledge on a variety of security subjects and is willing and able to research and learn about new technologies (and presumably, related regulatory requirements that may trail in their wakes). Navigating such a labyrinthine maze of options is a daunting task even for a well-schooled and seasoned Chief Security Officer (CSO), Chief Technology Officer (CTO), or IT administrator. IT security involves complex subject matter, including numerous specific network protocols and convoluted security regimes, plus access rights and permissions for systems across multiple domains and geographically dispersed organizational assets.

HIPAA imposes national standards for securing and maintaining privacy and security for protected health information (which includes medical data and other personal information such as name, address, and so on). The Privacy Rule applies to protected health information in all forms; the Security Rule applies to electronic data. This makes it imperative that all types of information, electronic or not, are safeguarded appropriately. The HIPAA standard encompasses three defined types of organizations: health care providers (for example, clinics, hospitals, and pharmacies), health care insurers, and health care clearinghouses.

GLBA, or the Financial Services Modernization Act of 1999, applies to financial institutions and organizations that deal in third-party monies such as banks, brokerage firms, and consumer credit reporting agencies. Credit counseling services, debt collection agencies, real estate services, and income tax preparers are also regulated by GLBA. GLBA comprises three portions, only one of which applies to IT professionals—the Safeguards Rule. Section 501 of GLBA addresses “Protection of Nonpublic Personal Information,” which requires that financial institutions adhere to administrative, technical, and physical safeguards for handling customer information. The Safeguards Rule section governs the collection and disclosure of sensitive client financial information and requires that regulated companies specify a person or group responsible for GLBA compliance, identification of security risks, safeguard assessment, safeguard implementation, security monitoring, compliance assurances, and completion of necessary upgrades to maintain compliance.

SOX sections 302 and 404 are the portions that affect IT departments for regulated companies. All companies registered under the Security Act of 1993 must comply. Under these sections, companies are required to establish an infrastructure secure from unauthorized data access or alteration, electronic damage and loss. This also means documenting all security measures taken.

Other relevant legislation includes California Senate Bill (SB) 1386, which went into effect on July 1, 2003. This bill requires all state agencies, persons, or businesses that conduct business in the state of California that also own or license computerized data (including personal information) to disclose any breach of security of that data as mandated in various ways by the bill. What is interesting about this legislation is that it requires reporting of any and all security breaches involving access to “unencrypted personal information” by unauthorized parties. Thus, in effect, this legislation not only mandates reporting of breaches in keeping with other similar regulatory requirements at all levels of government but also explicitly recognizes that encrypted data is protected and need not be reported should it be accessed by outsiders. This provides yet another strong argument for the value of encryption for data, either when stored at rest or in transit during file transfer, transaction processing, email, and so forth.



For more information about SB 1386, please visit http://info.sen.ca.gov/pub/01-02/bill/sen/sb_1351-1400/sb_1386_bill_20020926_chaptered.html.

Third-party services and vendors can be recruited where in-house technical knowledge or resources are scarce to none. Various storage vendors provide assessment services from auditing a storage environment for security weaknesses to drafting open solutions to meet business-specific needs and standards. Their data storage network experts can assist in the evaluation of services and products tailored to fit a given organizational structure and data storage requirements. Such services take a holistic life cycle approach with coverage through a four-cycle process: risk assessment, prioritization, implementation, and monitoring.

Stick to Security Standards

If there is a moral that emerges from the preceding section, it provides the title for this one—namely, that working with well-documented and well-understood industry standards remains an enterprise's best hope for establishing and maintaining security. Though it's tempting to view Andrew Tannenbaum's often-cited observation, "The nice thing about standards is that there are so many to choose from," with a certain amount of irony, a strict focus on established industry security standards nevertheless offers the best (and some would argue, only) chance of implementing security tools and controls that implement security policy correctly and conform to applicable laws, regulations, and best practices. To that end, a thorough understanding of TCP/IP security protocols and services comes highly recommended and should include working knowledge of IETF and W3C standards and protocols for secure transport, secure sockets, secure encryption and authentication, PKI, distributed applications, and secure remote access. This represents a huge body of knowledge and is probably best approached by seeking out experienced, knowledgeable individuals who have worked with and around at least some of these technologies for some time. It may also be useful to think about finding people who are certified with credentials such as Certified Information Systems Security Professionals (CISSP), Certified Information Security Managers (CISM), Cisco Certified Security Professionals, or even Cisco Certified Internet Experts (CCIEs) from the security track.

Making a Technology Selection: File Transfer Topology

The type of connections among sites is what determines network topology in an enterprise; this in turn dictates how file transfers occur. Although many organizations employ a mixture of types, common network topology models include hub-and-spoke and partial mesh organizations. A hub-and-spoke model arranges satellite sites around a hub (like a wheel, where spokes link the central hub to various locations on the "rim"), where traffic between satellites must pass through the hub to get from one to another (more commonly, large enterprises might have multiple hubs, each with its own spokes, and high-speed links between hubs, or perhaps even a hierarchy, with a central hub at headquarters connected to distribution hubs, each with its own regional spokes). A complete mesh is a topology whereby every node in a graph is connected to every other node, and reflects a situation where every site is connected to every other site in an enterprise network. This is prohibitively expensive once the number of sites goes much above half a dozen, so the reality is that sites that need links (or that are geographically closest) to one another get connected, and sites that communicate infrequently use Internet links to access otherwise inaccessible sites.

Hub-and-Spoke Transfer Environment: Pros and Cons

Many large organizations employ a traditional hub-and-spoke model to network computer systems. In fact, basic data transfer protocols such as HTTP and FTP are inherently hub-and-spoke by nature, as a spoke needs to initiate contact and possibly handle specific rules for login credentials, file names, and so forth.

As a networking model, the hub-and-spoke methodology is characterized by hub locations at corporate and regional headquarters or a single data center, with many point-to-point spoke connections extending out to branch offices. Using telephony as an illustration, each phone line represents a spoke where each phone company point of presence represents a hub. Ease of implementation, efficient cost allocation, and simple management are primary motivations for using hub-and-spoke configurations.

With health care systems, hub-and-spoke arrangements were the only initial way to create connections, where a subscriber establishes listening technology with a provider serving as the spoke to automate dial-up technology. A large subscriber company could potentially establish thousands of connections with no real unified management for these connections, as no all-purpose standard protocol exists in the health care industry. Small subscriber spokes are forced to tailor their systems to larger companies of importance, and it's not possible for a provider to tailor its communications scripts for all subscribers owing to a lack of uniformity. This drawback also impedes hub-to-hub and spoke-to-spoke communications, so that specially configured spokes are difficult to impossible to support using automation software.

As application needs and usage patterns transition into more collaborative knowledge-sharing trends, it becomes necessary for organizations to rethink this model. Yielding higher end-user productivity—especially through collaborative efforts and global pooling of organizational resources—will exercise the hub-and-spoke paradigm in ways it may not adequately handle. Multi-Protocol Label Switching (MPLS) and VPN technologies are two proven alternatives to the hub-and-spoke design that vastly improve upon the performance and capabilities of traditional point-to-point topologies without requiring a plethora of WAN links to interconnect multiple sites.

Finding the Best Fit for Your Environment

Each environment introduces its own challenges to achieving regulatory compliance, but the fundamental principles remain largely the same. The following checklist briefly describes these principles:

- Authorization and access should remain consistent across the infrastructure irrespective of the implementations and frameworks used to exercise these controls.
- Data retention and retrieval procedures should also remain consistent, in that information should be restricted only to authorized parties where it gets stored and obtained.
- Accountability can be maintained any number of ways, from platform-specific software monitoring components to general-purpose traffic analysis, intermediate network appliances, or distributed measurement facilities.

Reworking Automation

Mission-critical software, like mission-critical hardware, should be made to operate in a cyclic and continuous manner. There are far too many types and classifications of network-driven events that occur on a regular basis for any one person or group of persons to maintain and manage. Endpoints change or may come and go at a whim for mobile users, topologies may change due to impaired routing conditions, and threat levels are dynamic and therefore constantly change. Assessing, evaluating, and documenting these changes for compliance purposes can be exhausting as is, so automated batch processing of the most mundane and easily reproducible tasks should be a natural extension of any existing IT administration process.

Scripting, Scheduling, and Batch Processing

Task queues are essential for provisioning and managing modern computer systems. Even workstation and notebook systems are task oriented nowadays and are capable of scheduling services to operate at specific times or at regular intervals. Multiple similar tasks may occur concurrently and in parallel as schedules dictate, and run scripted or unscripted, tailored to meet site-specific needs and situations. At least some of these tasks are likely to involve file transfers of some kind, for everything from new commands, software updates, antivirus or anti-spyware signature files, directory updates or replication, backups and archives, and so forth.

Just about every platform from workstation to server has integrated scheduler capabilities and facilities for batch processing tasks. These automated chores take away most of the mundane administrative tasks that would otherwise be the responsibility of IT staff and end users, which is especially ideal for large-scale networking environments. UNIX and Windows hosts in particular ship with native scripting and scheduling facilities for a variety of automation tasks and purposes. Take advantage of these properties as they apply to creating a well-maintained business computing environment.

Summary

This chapter covers a lot of ground from scoping the network environment to enumerating organizational assets. Security is a process, not a product. There is no one-shot, cure-all, set-it-and-forget-it solution. The security and compliance process requires astute attention to detail, due diligence, and an unflinching persistence. Security is a deep and complex issue that is difficult to get right, even the second and third time around.

The following list summarizes the content covered in this chapter:

- Start by inventorying organization assets. Take stock of all server hardware, software, services, and protocols within the network environment.
- Identify types of file transfer protocols and classify data exchange-based services. Create a baseline of known-good services and protocols and take note of any network anomalies.
- Consult with compliance management officers, teams, and third-party professionals to obtain an accurate view of your business-specific needs. Ensure that the rules of compliance are clearly defined and strictly followed in practice.
- Analyze the IT infrastructure for risk and vulnerability and document every security implementation used to reduce risk factors for each case.
- Seek insecure file transfer solutions where PII is at risk of exposure to unauthorized parties. Legacy applications with no suitable drop-in replacement should be secured as well, perhaps by creating a cryptographically secure network connection between endpoints.
- Interoperability between platforms, protocols, and services is instrumental to maintaining consistency and reliability. One secure solution may be implemented differently on a Windows host versus on a UNIX host, but they should interact in an identical fashion.
- Verify and validate everything. Though a software protocol, package, or platform may advertise itself as undefeated or unbreakable, it must be validated and verified by auditors, penetration testers, or other security professionals time and time again.
- Automate everything possible. In doing so, you relieve strain from IT staff having to perform repetitive or routine tasks that are better handled by integrated system scheduling services.

Chapter 3: Securing File Transfer

Through earlier chapters, the impetus behind securing any and all forms of at-risk file transfers was clearly elucidated and loosely enumerated. Now that you know the answers to *why* you should secure at-risk file transfers, you will discover *how* such transfers can be secured.

This chapter begins with a topical discussion of network reference models, protocols, and services to describe the security concepts that are introduced immediately thereafter. The chapter takes a ground-up approach to describing the aspects of securing digital communications in the context of public networks. A discussion then follows with several worthy and practical solutions to establishing and maintaining secure communications.

Request for Comment (RFC) documentation is far and away the best jumping-off point for most of the material referenced in this chapter. This series of documents encompasses research, insight, and applicable methodologies related to Internet technology, and serves as the basis for drafting and formalizing that terminology, key concepts, and recommended implementations. Because they are such excellent resources for further exploration of the topics described, RFC references are cited for many of the concepts discussed in this chapter.

TCP/IP and Network Reference Architectures

Before digging into specific TCP/IP security architectures, such as IP Security (IPsec), let's first make a cursory foray into the Open Systems Interconnect (OSI) basic reference model, in which all TCP/IP-based protocols and transactions can be identified and classified according to one of its seven layers of operation. For completeness, Figure 3.1 also shows the original TCP/IP networking model, which consists of only four layers.

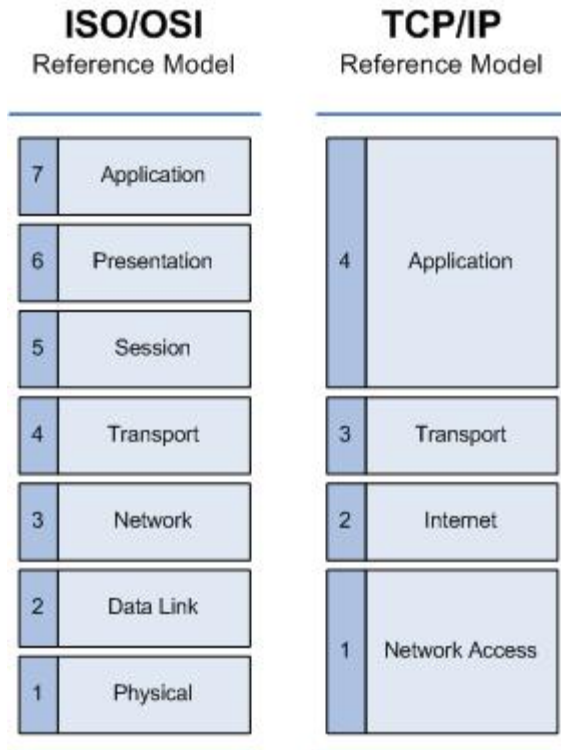


Figure 3.1: The ISO/OSI basic reference model describes network protocol interaction in layers.


The biggest benefit of a layered approach to network communications is that it takes a large, complex, end-to-end problem (network communications) and breaks it up into a series of inter-related but less complex issues that may be solved independently and with less overall effort. Because this has the additional benefit of decoupling hardware signaling and communications development from software development, it's pretty much driven network research and development from its inception, even before this widely used model was spelled out in the early 1980s. Each higher-layer protocol depends on services from the lower layers that encapsulate them as information is passed up and down this network stack.

The ISO/OSI model layers may be summarized as follows:

- Layer 7: Application Layer—A user interface to access information across the network through application protocols. Such uses include SSH, FTP, and HTTP.
- Layer 6: Presentation Layer—Transforms data from lower layers into formats readable by the application layer, and vice versa. Such uses include data compression, encryption, and encoding.
- Layer 5: Session Layer—Controls sessions between network endpoints by establishing, maintaining, and terminating communication among hosts.
- Layer 4: Transport Layer—Provides a transparent transfer of data between endpoints through flow control, segmentation, and error control. TCP and UDP reside at this layer.
- Layer 3: Network Layer—Supplies functional and procedural methods of transmitting variable-length data sequences, performs routing functions, enforces quality of service (QoS) policies, and handles segmentation.
- Layer 2: Data Link Layer—Supplies the functional and procedural means to transfer data and correct errors occurring on the lower level. Ethernet and Media Access Control exist on this level.
- Layer 1: Physical Layer—Consists of electrical specifications and physical provisions in the form of network devices or interfaces. Physical Ethernet standards are in this layer.

The TCP/IP network reference model actually preceded the development of the ISO/OSI network reference model and describes the more common ways in which networking protocol software is implemented inside or alongside most modern operating systems. In the TCP/IP reference model, the capabilities of the OSI/ISO Session and Presentation layers get bundled into the Application layer, and capabilities at the Data Link and Physical layers likewise are subsumed in the Network Access layer.

A packet is the basic atomic unit upon which all TCP/IP-based transmissions are built. Packet construction, usage patterns, and protocol capabilities characterize the type of transmission that occurs for a given connection. There are essentially two direct ways to build, send, or receive and process file transfer-capable packets: streams and datagrams. Although there is the distinct possibility of transferring data within the misused or unused header portions of TCP/IP and ICMP packets, its usage is unorthodox and ill-advised for these purposes, appearing mostly in covert channel communications between attacker and victim. As such, the subject will not be discussed further.

 There are many excellent technical write-ups available on the subject of covert channels embedded by data hiding in the TCP/IP protocol suite. Two such examples include http://www.firstmonday.org/issues/issue2_5/rowland/ and a paper titled “Covert Channel Analysis and Data Hiding in TCP/IP” at <http://gray-world.net/papers/ahsan02.pdf>.

Streams-based protocols are used in a variety of ways, but they all ensure and refer to a succession of elements made reliably available over some period of time through TCP. A streaming communications channel provides a seemingly endless flow of information between sender and recipient. The datagram approach is oriented toward lightweight delivery of individual parcels and not concerned with session management, unlike the TCP streams-based approach. In effect, you get faster transmissions with UDP because it lacks the performance-deteriorating overhead of streaming channels.

What Is Tunneling?

Tunneling is the process of encapsulating a network protocol or session within another presumably lower-layer protocol. Previous chapters introduced the terms Virtual Private Networking (VPN) and Multi-Protocol Switching Labels (MPLS). Some VPN solutions utilize what is called the Layer-2 Tunneling Protocol (L2TP) for secure connection establishment, while MPLS itself is a tunneling protocol that operates in an abstraction between layers 2 and 3 of the OSI reference model. A VPN effectively creates a tunnel for site-to-site communications over public channels.

Tunneling permits unusual and creative new uses and reuses of existing protocols, services, and sessions. Although not all tunneling protocols are security related, many of them do permit some form of connectivity that is otherwise administratively or technologically prohibited or unsecured in its native form. Take, for example, protocols 6to4 and SSH—where 6to4 tunnels IPv6 traffic into IPv4 channels, SSH can tunnel the largely insecure FTP using the SSH protocol as a secure encapsulation mechanism.

The following items are all examples of tunneling protocols:

- Point-to-Point Tunneling Protocol (PPTP)—A dual-session protocol designed to encapsulate protocols in VPN configurations; actually uses GRE.
- Point-to-Point Over Ethernet (PPPoE)—Used to encapsulate Point-to-Point Protocol (PPP) frames within Ethernet frames as with Asynchronous Digital Subscriber Line (ADSL) uplinks.
- IP in IP (RFC 1853)—A method of tunneling IP within IP packets, whereby the first IP header contains protocol data specific to the tunnel with another IP-based header as a payload.
- 6to4 (IPv6 over IPv4)—A means to encapsulate IPv6 packets to traverse IPv4 networks such as the Internet, without the need to configure explicit tunnels.
- Secure Socket Layer (SSL)—An application-layer protocol used for the management and secure transmission of private information via public networks.
- Transport Layer Security (TLS)—The successor to SSL; serves an identical purpose and shares much of the same features and capabilities as SSL.
- Secure Shell (SSH)—A set of standards and an associated network protocol for establishing secure connections between endpoints.



Note that the last three protocols SSL, TLS, and SSH are actually stream-based protocols while the rest are datagram-driven.

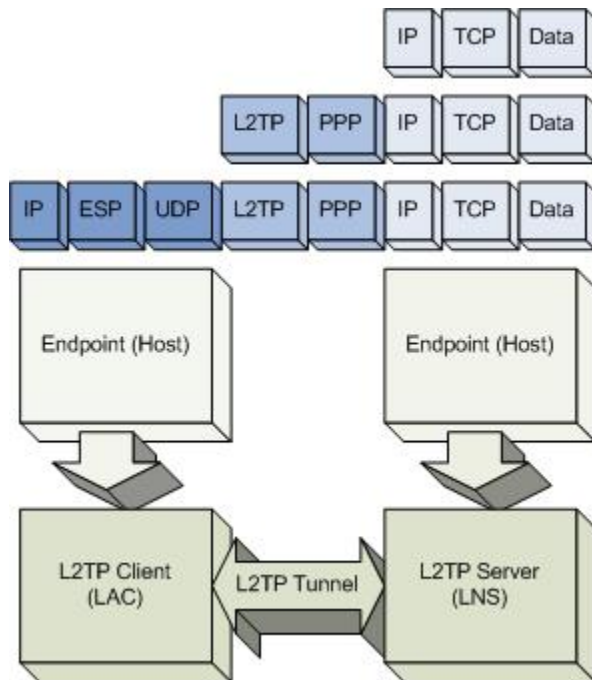


Figure 3.2: How tunneling works using L2TP.

What Is IPsec?

IP security (IPsec) is a suite of protocols for securing Internet Protocol (IP) transmissions between unprotected and protected interfaces for a host or an entire network of hosts. IPsec lays the foundation for authentication, encryption and protocols for cryptographic key establishment for use with connection-oriented data streams (TCP) or unreliable datagram transactions like the User Datagram Protocol (UDP).

As the name suggests, IPsec is a layer 3 protocol in the OSI reference model (in metaphorical terms, it sits between layers 2 and 3 as a kind of gatekeeper and guardian). While this aspect gives strength to IPsec as a mechanism for encapsulating unsecured transmissions, its use also introduces increased implementation complexity and computational and communications overhead because IPsec cannot leverage TCP (layer 4) for management and reliability functions.

IPsec is no turnkey solution in and of itself, but rather a single modular portion in a much larger, more complex security equation. Three other site and user-specific components of that equation include appropriate security protocols, cryptographic algorithms, and encryption keys.


Why Is IPsec Used?

IPsec provides transparent encryption and security services for IP network traffic as an add-on option in IPv4 and as a required element in IPv6 environments. Therefore, IPsec effectively bridges the gap left by an original omission of security provisions in the initial implementation of the standard Internet Protocol (IPv4). Traffic traversing the boundary of an IPsec-controlled perimeter is subject to administrative access controls that specify whether packets should be passed unaltered, discarded, or be wrapped in additional security services. VPNs can be and commonly are established, managed, and maintained using the IPsec protocol suite.

The following security services are made possible through the IPsec framework:

- Traffic encryption to deter eavesdropping
- Integrity validation to thwart tampering
- Peer authentication to secure conversations
- Anti-replay protection against playback attacks

The security design goal for IPsec is to provide cryptographically secure interoperability between devices and to ensure confidentiality, party verification, integrity checking, and rejection of attempts to resubmit previously recorded traffic to gain unauthorized access (playback attacks).

 A playback attack occurs when the time-sensitive bits contained in previously recorded network traffic are altered and resubmitted to a target network in an attempt to fool a server and service into thinking this falsified data is part of some existing and/or legitimate traffic. Although such instances are unlikely, they are possible under certain circumstances and merit enough attention to warrant proactive countermeasures for at-risk servers.


As a protocol suite, IPsec comprises the following components:

- Security protocols—Authentication Header (AH) and Encapsulating Security Protocol (ESP)
- Security associations—A simple connection context that provides security services to the traffic it carries.
- Key management framework/standard—For distribution, management, and utilization of public keys for the authorized operation of security services and protocols.
- Algorithms for authentication and encryption services.

The next section describes IPsec usage in basic detail with pointers to more specific documentation.

How Is IPsec Used?

Two distinct protocols lay the foundation for network-based security services: the optional AH and mandatory ESP, described in RFCs 2402: IP Authentication Header and 2406: IP Encapsulating Security Payload, respectively.

 See <http://www.ietf.org/rfc/rfc2402.txt> for a full description of the AH and <http://www.ietf.org/rfc/rfc2402.txt> for details on ESP.

The AH protocol offers both integrity and authentication capabilities with optional anti-replay features at the receiver's discretion.

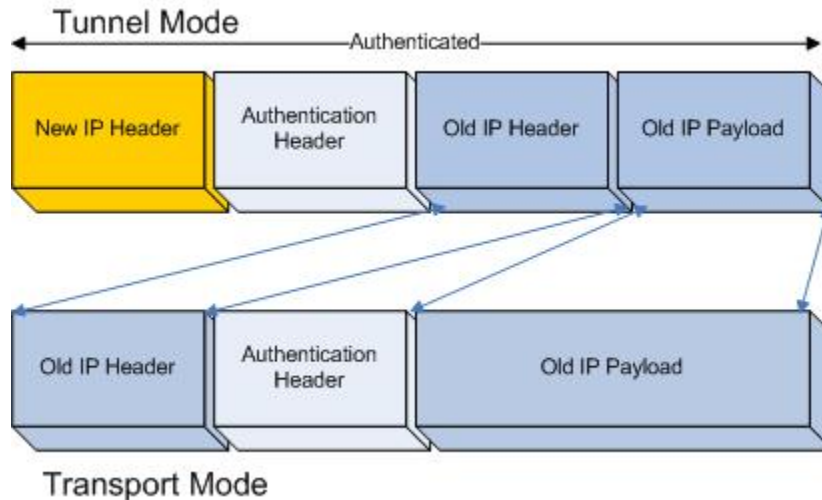


Figure 3.3: The IPsec AH protocol format.

ESP offers all three features plus confidentiality. Though ESP can be used to provide only integrity without confidentiality, ESP use with confidentiality and no integrity is not recommended.

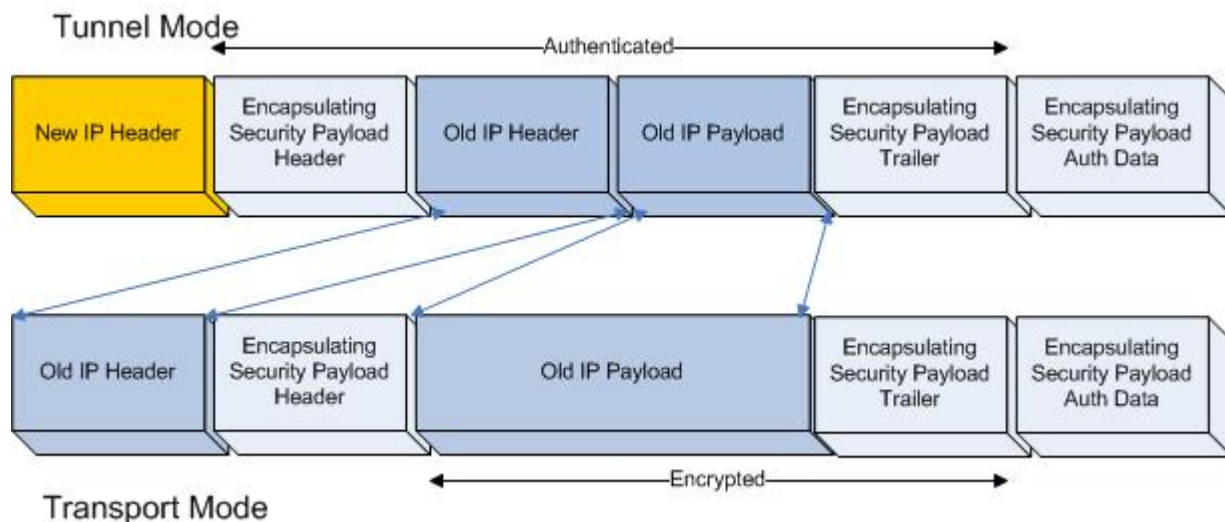


Figure 3.4: The IPsec ESP format.

Both AH and ESP provide access controls enforced through distributed keys and traffic management governed by a Security Policy Database.


Service coverage includes interoperation between endpoints in one of three ways:

- Host-to-host
- Host-to-gateway
- Gateway-to-gateway

Fine-grain control of security services makes it possible to establish a single all-purpose gateway-to-gateway or individualized TCP tunnels for host-to-host connections. IPsec supports two separate encryption modes: transport and tunnel. Transport mode provides end-to-end security for traffic, which requires that endpoint computers handle security processing, but encrypts only the payload portion of each packet leaving the header untouched. Tunnel mode provides point-to-point communications security through which many client computers share a single gateway node and this process encrypts both header and payload content.


Sending and receiving devices are required to share a public key to interoperate under the auspices of IPsec. Most of the security services provided through IPsec require cryptographic keys and therefore rely on separate mechanisms for key exchanges. The RFC 4301: Security Architecture for the Internet Protocol describes a specific public key approach for automated exchanges and states that other key distribution techniques may be used.

Internet Key Exchange (IKE) is the protocol of choice for many IPsec implementations and is the de facto standard for security association establishment. IKE uses a Diffie-Hellman key exchange for public or pre-shared secret keys to mutually authenticate communicating parties. The Internet Security Association and Key Management Protocol (ISAKMP), explained in RFC 2408, forms the basis of IKE. Menezes-Qu-Vanstone (MQV) is another authentication protocol for key agreement based on the Diffie-Hellman scheme. Other asymmetric key algorithms include Rivest, Shamir, and Adleman (RSA), Elliptic Curve Cryptography (ECC), El Gamal, the Digital Signature Algorithm (DSA), and Knapsack.

 IKEv1 is defined in RFC 2407, RFC 2408, and RFC 2409; IKEv2 is defined in RFC 4306.

IPsec may be integrated into endpoint hosts or devices, intermediary devices themselves, or specialized standalone network security appliances placed inline. The concept of IPsec is fully elaborated in the corresponding RFCs, while the subject remains a deep and complex practice for implementers and designers.

Earlier implementations of Network Address Translation (NAT), a means of mapping many non-routable addresses to a single or few mutually shared routable public IP addresses, would fail to pass IPsec traffic. When IPsec traverses one of these preliminary NAT interfaces, a mission-critical hash value is altered, rendering the IPsec provisions useless. A method of safely encapsulating UDP-based IPsec messages is defined in an IETF draft. Modern routing appliances and network devices inherently supporting NAT will harmlessly pass IPsec traffic, but there may be instances where you encounter the resistance of an older implementation.

 UDP encapsulation of IPsec packets is explained at <http://www3.ietf.org/proceedings/04mar/I-D/draft-ietf-IPsec-udp-encaps-08.txt>.

What Is Secure Socket Layer?

SSL is an IETF-approved standard that provides secure communications schemes for sharing private data across the public Internet infrastructure using public key encryption. It operates at the transport layer and up (layers four to seven) in the OSI reference model, so SSL lacks the flexibility, increased processing, and added complexity of IPsec.

SSL is characterized by and comprises the following security services:

- Message confidentiality
- Message integrity
- User authentication
- Key exchange services

Four distinct mechanisms make these services possible. Message confidentiality is made possible through symmetric ciphers or shared-secret key cryptography. Message integrity is provided through message digests or calculated hashes using the encrypted data as input and a fixed-length checksum output. Authentication and key exchange, both being separate components, are established through the use of an RSA handshake, where digitally signed certificates and session keys are exchanged.

Digital certificates are issued by well-established Certificate Authorities (CA), either corporate internal CA or external CA(s) such as VeriSign. These certificates contain the following:

- Digitally signed identifying information for both subject and issuer
- A range of temporal validity
- The subject's public key

It is this final item—the public key—that is the core component of the RSA key exchange. When paired with its counterpart, the private key, two parties may begin sharing information in confidence. For all the relative strengths, security merits, and privacy assurances that an asymmetric key exchange like this offers, it is also highly computationally expensive to conduct in large network environments transacting upon many simultaneous multiple interactions, especially for high-traffic areas servicing thousands within a small window of time.

Why Is SSL Used?

The SSL protocol permits client and server applications to communicate securely by hindering or deterring the following tactics:

- Eavesdropping—The receipt of a message by unauthorized and unintended parties
- Tampering—Unauthorized access and undesirable modification of a message
- Forgery—A certified message sent with deliberately falsified information

Typically these actions will occur one after another against one or more elements of data, perhaps in a man-in-the-middle (MitM) attack pattern. If an unauthorized party or unintended subject has tampered with a message, it is implied that this individual or group also eavesdropped and thereby intercepted the message. Likewise, to commit forgery implies both eavesdropping and tampering, for without both intercepting and deliberately modifying the message, there is little—if anything—malicious that can happen to the message. Prevent eavesdropping and you stand a good chance that data will pass without tampering. Prevent tampering and you ensure a great deal of resistance to forgery.

SSL is used in many places and is often transparent to the end user. Web browsing, email, and Internet-based faxes are three common instances in which SSL is frequently used.

How Is SSL Used?

During the transfer phase, SSL sends an optionally compressed record that is encrypted and packed with a message authentication code (MAC) and specifies what upper-layer protocol is encapsulated. Public key cryptography operations closely associated with the SSL handshake process occur at the start of every SSL-based connection. It is this crucial moment that impacts performance for a multi-function, general purpose Web or FTP server.

Account and password pairs often appear where secure terminal access or centralized account sign-on services are used. There may be other means, perhaps through keys or smartcards, but the point is that a user is authenticated in some fashion to gain access to a certain network application.

Through SSL, the server will negotiate a secure channel by sending a message to the client indicating a desire to establish cryptographically secure communications. The client application will then declare its security parameters, which the server compares against a set of its own in search of a match. This process is called the SSL handshake.

The server authenticates to the client by sending a digital certificate, and at the client's discretion to trust the server's identity and certificate, will continue or discontinue communication. The server can in turn require that the client issue a digital certificate for mutual authentication purposes during this phase, but this is not always the case. The client generates a session key, encrypts it with the server's public key, and issues it back to the server. Both parties then use this symmetric key to encrypt data passed between, establishing a secure channel for the duration of that session.

What Is TLS?

TLS is a proposed Internet standard intended to succeed SSL; it is also a cryptographic protocol that provides secure network transmissions across potentially hostile media, such as the Internet. Though slight and subtle differences exist between SSL version 3.0 and TLS version 1.0, both protocols and their usage constraints remain much the same. In practice, the terms are used interchangeably; however, a common misconception is that TLS is the *same* as SSL, which it is not. Though derived from SSL v3.0, TLS is a different protocol altogether and their differences are substantial enough that the two are not interoperable.

 For more information regarding TLS, see <http://www.ietf.org/rfc/rfc2246.txt>.

Why Is TLS Used?

The primary goal for TLS deployment is to provide privacy and data integrity between applications, just like its predecessor SSL. TLS is a separate protocol from SSL, but they share the same design goals. It, too, is a two-layered composition consisting of a TLS Record and TLS Handshake protocol layered atop a reliable transport protocol.

The TLS Record protocol exists at the lowest layer, stacked above a reliable transport protocol, such as TCP, and is used to securely encapsulate various upper-layer protocols. It provides two essential connection security needs:

- The connection is private
- The connection is reliable

The TLS Handshake protocol establishes a secure communications channel between endpoints, negotiates encryption algorithms, and exchanges symmetric keys. This is necessary before the application-layer protocol can transmit or receive any amount of data. It provides the following security elements:

- The peer's identity can be optionally authenticated
- The negotiation of shared secrets is secure
- The negotiation is reliable


How Is TLS Used?

TLS operates beneath application protocols such as HTTP and FTP and above TCP or UDP. Like SSL, TLS appears primarily in Web-based merchant or electronic commerce sites, online retailers, and banking or financial institutions—wherever security is particularly necessary. TLS can be utilized as a tunnel through application wrapper means such as Stunnel or an entire network stack in the form of a VPN, much like OpenVPN does. The free Stunnel SSL/TLS service provides tunneling for those clients and servers that may not otherwise support such capabilities on their own, while OpenVPN is a free, full-featured VPN built on the OpenSSL library.

As an interesting side note, there are many hardware-assisted solutions, often referred to as *cryptographic accelerators*, for offloading resource-intensive computing tasks associated with cryptography. Instead of burdening the general purpose CPU with the heavy-lifting tasks of cryptographic processing, the task is delegated to a more specialized processing component.

What Is SSH?

SSH is a protocol originally developed in 1995 by Tatu Ylönen from the Helsinki University of Finland in response to a password-sniffing attack of that institution's network. The original impetus behind SSH was to replace the rlogin, TELNET, and rsh protocols, which neither used strong authentication nor provided confidentiality for data transported. This first version of the technology became known as SSH-1 and is served by free and open implementations like those available at <http://www.openssh.com> as well as commercial (and more powerful) implementations like those offered by SSH Communications Security. SSH-1 ultimately evolved into SSH-2, an Internet standard covered in the secsh group of RFCs (documented in RFC 4251, which describes the SSH-2 architecture).

 You can access RFC 4251 through its sponsoring body, the IETF, at <http://tools.ietf.org/html/rfc4251>. Or check out the RFC references from O'Reilly's *SSH: The Secure Shell (The Definitive Guide)* on its "SSH Protocol" page at <http://www.snailbook.com/protocols.html>.

SSH consists of a set of standards and an associated network protocol that permits a secure channel to be established between a local and a remote host that uses public-key cryptography to authenticate the remote host (it also includes optional provisions to enable the remote host to authenticate the user as well). SSH protects message content and ensures message integrity through a combination of encryption (to ensure content confidentiality) and message authentication codes (MACs, to ensure content integrity).

How Is SSH Used?

The most typical use for SSH is to enable users to log into a remote host to execute commands, in much the same spirit as the original command-line utilities (rlogin, TELNET, and rsh) it was in part designed to replace. But SSH also supports tunneling, can forward X11 connections or TCP ports on a user's behalf, and can transfer files using either Secure FTP (SFTP) or the Secure Copy Protocol (SCP). SSH is associated with standard TCP port 22 (ironically, adjacent to ports 20 and 21, which are associated with FTP). SSH offers nearly the same capabilities and functions as both IPsec and SSL/TLS but is considerably less vexing to deploy and less challenging to maintain.

Using VPNs

Ostensibly, a VPN is used to secure private communications over non-public network media. VPN traffic is shuttled via the public networking infrastructure atop standard protocols or through privatized leased-lines between customer and service provider. A VPN can establish internal, site-to-site, inter-organizational, or inter-territorial connections for confidential communication for one or several companies.

Benefits of a well-designed multi-site VPN include:


- Expanded geographic or territorial connectivity
- Improved site-to-site security
- Reduced operational costs versus Wide Area Networks (WANs)
- Reduced transit time/transportation costs for mobile employees
- Simplified network topology in some instances
- Global networking with telecommuter support
- Excellent scalability with Public Key Infrastructure (PKI)

VPNs may utilize public or private telecommunication infrastructures; what remains constant is their use of tunneling protocols and security procedures or processes to achieve confidentiality between endpoints. Encryption plays a full-time role in this capacity, and compression occasionally makes appearances in supportive roles.

IPsec VPNs

IPsec is commonly used in the development and implementation of many VPN solutions. It provides a virtual conduit through which two networks can share resources as one, opening the door to external client callers so that they may access private internal resources. An IPsec implementation is ideal for transparent high-performance passageways between organizations that may be territorially or regionally separated, wherever general purpose coverage applies. In some cases, IPsec is integrated directly into the kernel or inline appliance network stack, and can be facilitated by specialized hardware designs.

Do not be fooled by the misconception that IKE-based VPNs are invisible on the network. Transparency is highly desirable for any type of networking connection; that is why a VPN should operate with little visibility to the end user short of status and link-state indicators. Discovery of IKE systems is made possible through tools such as NTA Monitor's ike-scan tool. This tool exploits characteristics in IKE via specially crafted probe packets that stimulate an IKE response and can, in some instances, identify a VPN solution down to vendor make and model.

 Obtain a full-length description or download ike-scan from <http://www.nta-monitor.com/tools/ike-scan/>.

Implementation Approaches

IPsec VPNs are ideal where high-speed, redundant, site-to-site connectivity is necessary. An IPsec solution is designed to meet the criteria for security provisions within always-on network accessibility contexts for ensured confidentiality and enhanced productivity. A VPN based on IPsec gives geographically dispersed regions a means of mutually secure interconnection and the ability to securely service mobile or off-site clients in an on-demand fashion.

However, with these accommodations come many technical considerations. IPsec is costly to employ both in terms of software and any requisite hardware, most of which is vendor-specific. Also, there is no guarantee that one IPsec solution is interoperable with another, and some networking devices providing NAT services may modify IP addresses in a manner unsuitable for IPsec use. This is particularly true when public IP addresses are required at both ends of an end-to-end connection, and NAT is used at one end or the other, specifically to mask the source or destination IP address (or to permit private IP addresses to access the public Internet).

IPsec is the logical and perhaps most cost-effective choice for establishing remote or branch office connectivity. Due to the specialized components, IPsec ties the end user into using a single computer with individualized site-specific configurations and firewall considerations. However, IPsec VPNs provide neither the easiest nor the most cost-effective method for securing file transfers internally, or even file transfers between partners. Setting up inter-company VPNs can be cumbersome and is seldom trivial, so unless the companies exchange large volumes of data, this may not be advisable for file transfers. Even for internal use, it can be difficult and cost-prohibitive to use IPsec VPN software, so other solutions—most notably, those built around SSH—are often selected to provide true end-to-end security.

SSL VPNs

SSL/TLS VPNs are typically easier to install, support, and maintain than their IPsec counterparts. Because they operate at a higher layer than IPsec, the SSL/TLS variety can provide better granular access control required by remote access and extranet VPNs. An SSL-based VPN delivers user-level authentication to ensure that only authorized parties have access to authorized resources, and the competitive advantage of on-demand access anywhere due to the ubiquitous presence of SSL.

SSL traffic also tends to pass relatively unimpeded across the network, where IPsec sometimes encounters uncooperative NAT devices that can impede IPsec traffic because they hamper the formation of end-to-end connections (but which may be impossible anyway on networks that use private IP addresses). And because SSL is integrated into many common applications, such as popular Web browsers, there is not necessarily the need for vendor-specific client applications on every participating computer.

SSL VPNs also tend to exhibit a lower cost of ownership compared with IPsec solutions with easier scalability and simple access for Web-enabled applications. User-level access controls are necessary as clients may call in from unknown and/or untrusted computers, where sensitive information may be left exposed on public terminals. Still, SSL VPNs are an emergent trend in the market space next to IPsec. Although an SSL VPN solution goes a long way toward reducing administrative and operational overhead, it may not provide everything that an IPsec-based solution does.

Implementation Approaches

The SSL solution can tunnel specific applications rather than giving more general access to the network segment, as is the case for IPsec solutions. This yields application-specific access to a network dictated by administrative controls. In addition, per-user permissions can be established with granular control. It also does not require site-specific configurations and the issuance of company laptops to each authorized mobile end user.

The IPsec approach generally requires third-party software and possibly their specialized hardware. This adds an extra layer of security by requiring each client to have a specific and specifically configured client application installed to use the VPN. Usually this also requires on-site services and financially costly per-seat licenses and per-person laptops for mobile users.

Each implementation expresses its unique advantages and disadvantages in different usage scenarios—never mind the convincing technical details. Where IPsec may create complications between communicating business partnerships, requiring that both agree on a vendor-specific platform and mutual configurations, SSL is much less demanding and ideal for mobile users just checking email. They are not, however, mutually exclusive technologies and can be used together to satisfy a variety of usage scenarios and client conditions.

SSH VPNs

The SSH protocol was not originally designed for use as a VPN, but as mentioned earlier, SSH may be and often is used to tunnel specific protocols end to end between pairs of computers. For example, you can use the SSH TCP port forwarding facility to tunnel the insecure FTP from one computer over an untrusted network to another computer. This provides a fully secure connection that neutralizes FTP's intrinsically insecure features (lack of encryption and authentication, plus lack of protection for user credentials and message content). Likewise, homegrown file transfer solutions or proprietary solutions such as those built around HTTP (usually to make them easily accessible to Internet users, all of whom have Web browsers) are also easy to secure with SSH without requiring any alterations to whatever programs may already be in use.

Implementation Approaches

SSH provides the same levels of security that other solutions offer—most notably, those based on IPSEC or SSL/TLS—but requires considerably less effort to implement, particularly because its use means that existing scripts, programs, and networking infrastructure need not be changed or replaced.

In addition, the SSH protocol may be transmitted across IPsec VPNs without requiring any additional work. This helps to explain the common belief that SSH is probably the most network-friendly security protocol around.

Normal practice is to set up SSH so that end-to-end security is provided. However, IPsec VPNs are most often used in gateway-to-gateway mode to provide a secure link from one network to another, whereas SSH typically provides secure links between a specific client and a particular server, thereby providing a more secure implementation.

Other VPNs

Although IPsec and SSL/TLS make excellent choices for a VPN, and SSH serves admirably for this purpose (particularly when the necessary client software elements are deployed to end users), they are not the only options available. PPTP, L2TP (including versions 2 and 3), MPLS, the Cisco-specific Layer 2 Forwarding (L2F) protocol and Multi-Path VPN are other usable forms of VPN technology. Some large ISPs include managed VPN services for business customers and take away the administrative overhead of operating such technology.

Securing FTP

At this point, you have an idea of both *why* you should use encryption and *what* can be used for this purpose. Securing FTP now becomes a question of *how*. The sections that follow offer many possible and practical solutions to at least partially answer this question. Ultimately the choices will be site-specific, calling upon existing IT knowledge and leveraging current infrastructure trends to adapt the network environment to a more secure workplace.

SSH

SSH provides a set of standards and associated network protocols to establish secure communications channels between endpoints, often across media that cannot be reasonably secured nor is guaranteed to be secure (such as the Internet). SSH uses public key cryptography for authentication, provides confidentiality and integrity for data exchanges, and utilizes message authentication for added tamper-proofing.

SSH was originally developed for secure systems administration on UNIX servers. But since its inception in 1995, it has increasingly been used to implement secure file transfer solutions as well. Today, many users worldwide employ SSH as a preferred method for secure file transfer on a wide range of computing platforms from Windows PCs to IBM mainframes.


The SSH protocol provides a set of standardized applications that creates an easy migration path to secure file transfer protocols and services. One key SSH-based application is SFTP, an application that is quite similar to FTP but that replaces FTP's insecure features with more secure alternatives. SCP is another key SSH-based application, modeled on the UNIX CP (Copy) command.

Users can choose either of these applications with confidence because SSH ensures strong authentication not only for individual users but also for the computers involved. Also all data transmitted between communication partners (including passwords) is automatically encrypted using advanced algorithms such as the Advanced Encryption Standard (AES) or Triple DES (aka 3DES, a mode of the Data Encryption Standard—DES—that encrypts all data three times over).

SFTP

SFTP is another network protocol and component of the primary SSH-2 design. This protocol facilitates secure network-based file transfer. In addition, the SFTP subsystem of SSH supports a complete set of file and directory operations, which makes it an entirely workable and reliable remote file system protocol.

SFTP shares many properties with FTP and supports all the latter's common capabilities, including directory listings, put/get file, file rename, and file deletion. Because SFTP is a subset (or subsystem) of the SSH protocol, it is platform independent and suitable for just about any computer that can operate an SSH client or server. Today, SFTP is available on virtually any computing platform from a Windows PC to an IBM Mainframe. SFTP is not, however, FTP simply shuttled via SSH; it is, in fact, a new and separate protocol designed from the ground up by Tatu Ylönen and standardized by the IETF SECSH working group (RFC4250-4254).

 Find more information about SFTP solutions based on the SSH protocol at www.ssh.com/solutions/applications/sftp.html and <http://www.ssh.com/products/efl/usage.html>.

Although SFTP has been available for more than 10 years, FTP is still widely used in many organizations despite its many drawbacks. There are many reasons why the use of FTP persists; it is clear that chief among them is that many organizations have built their applications and file transfer systems around FTP. Thus, moving away from this legacy is difficult and often requires considerable development effort. Some organizations have thousands of scripts to control file transfers on a daily basis: switching from insecure to secure implementations can thus involve immense effort to effect such changes.

Some vendors have created special software tools to help organizations make the transition to SFTP systems less difficult and resource consuming. Automated supports like these lower the burden of securing file transfer and make the conversion to secure systems significantly faster and cheaper.

SCP (Secure Copy)


SCP (Secure Copy), another part of the SSH suite, permits only file transfers. This makes it ideal for hands-off automated tasks and scheduled periodic bulk file transfers. SCP makes no provision for authentication or security but relies upon the underlying SSH protocol to handle these tasks.

During upload, a client supplies the server with content and can optionally include basic file attributes such as permissions and timestamps. FTP cannot handle these tasks. During a download, a client requests files or directories to be obtained from a server and the server does the rest, which turns the operation into a server-driven transaction.

FTPS (aka FTP/SSL)

FTP via SSL is yet another method for securing file transfers across the network. Information can be encrypted for the data channel, the control channel, or on both channels. When the data channel remains unencrypted, the protocol is said to be using a clear data channel; likewise, when the control channel is unencrypted, the protocol is said to be using a clear command channel. Interoperating FTP over SSL is defined in three ways:

- **SSL connect**—A connection is made to a separate port, typically 990 according to the Internet Assigned Numbers Authority (IANA), and the SSL negotiation is performed.
- **AUTH SSL**—A connection is made to port 21 in normal fashion, and AUTH SSL or AUTH TLS-P is issued to request SSL negotiation to implicitly protect any further transactions.
- **AUTH TLS**—A connection is made to port 21 in normal fashion, where AUTH TLS or AUTH TLS-C is provided in request for SSL negotiation to explicitly protect the ongoing data transaction.

 RFC 4217 fully describes the concept, operation, and procedures for FTP over SSL. Check out <http://tools.ietf.org/html/rfc4217> for more information.

Of these, the first two are deprecated and not recommended for usage, mainly owing to their lack of standards conformance. Implicit SSL protection of the FTP session (the first item in the previous list) occurs upon connection for control and data connections. The second method, AUTH SSL, permits a connection to negotiate a connection upgrade to SSL security. Once the control connection is secured, the data connection is secured, but the approach so thoroughly clashes with other RFC standards that its use is also not recommended.

The third option, AUTH TLS, is the method of choice according to RFC standards. It makes a client connection in typical fashion to port 21 on the target host and begins an unencrypted dialog with the FTP service. It requests that TLS security be used, performs the TLS Handshake process, and then issues any sensitive data over a secured channel.


Other Approaches

Current examples up to this point are relatively straightforward and intuitive: there exists a simple synergy between client and server applications with regard to previously described secure file transfer technologies. Other implementation approaches may seem unusual, unorthodox, and perhaps unnecessary for most uses. The next sections briefly investigate these remaining items, if only to give a pointer in one of the many tangential directions you may pursue.

Tunneling with SSH

SSH tunneling uses the concept of port-forwarding to achieve its goals. Highly restricted networks normally prohibit the passage of traffic to specific sites, specific ports, specific protocols, or some combination thereof. Port-forwarding helps to bridge the gap between some of these highly restricted destinations by establishing a gateway machine to act as a go-between. This machine, which could in practice be the same computer on which an SSH server is housed, brokers requests from client to server using whatever legal port pairing may be required.

A forwarding agent is established to listen on some presumably unrestricted port; for example, port 21. The forwarding agent is then directed to contact some given host—acme.com port 22, to continue this example—once a connection is established on port 21. This naïve illustration serves to demonstrate the basic nature of SSH tunneling but explains nothing of the more exotic arrangements and exciting scenarios to which such tunneling applies.

 See “Tunneling Explained” on the SSH Web site for more information about how to set up various tunneling, or port forwarding, arrangements using SSH (it’s available at <http://www.ssh.com/products/tunneling/usage.html>).

SSH tunneling is versatile and powerful and has been exploited by numerous software vendors. As described earlier, many organizations have performed file transfers for a long time and have invested heavily in setting up a file transfer infrastructure, often one that is based on FTP at least in part. Securing such legacy file transfers is important but can be resource intensive and costly.

The Push/Pull Concept

Understanding the push-pull concept is perhaps only marginally necessary to utilizing push-pull strategies and technologies in the workplace. After all, not every driver must possess an understanding of the four-cycle combustible engine to operate a motor vehicle. It is helpful knowledge, but not a prerequisite to operate the vehicle.

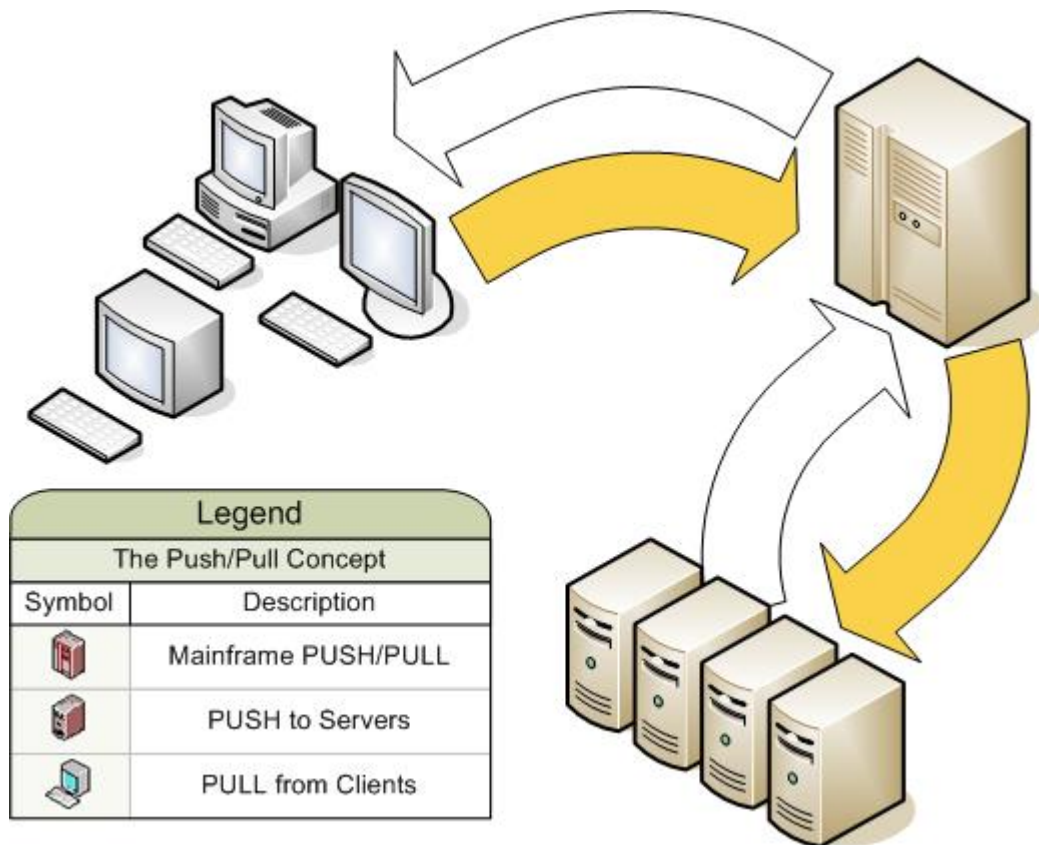


Figure 3.5: The push-pull concept illustrated here shows client computers pulling data from a mainframe computer also pushing data to servers.

In the *push* swing of the pendulum, the recipient need not request data transfer from the sender; the sender supplies what is wanted, perhaps even on a predetermined schedule. This might occur as some part of an automated mandatory update of integral software components, for example.

During the *pull* swing of the pendulum, the client requests a specific file or set of files to be transferred on *its* schedule, and pulls it through the delivery channel. This might occur as the client directly requests to have a software component updated.

Automated File Transfers

File transfer automation is the loyal servant of industrial technology. It is through scheduled automation that all routine and monotonous tasks may be handled consistently, correctly, and in conformance with a well-defined schedule. The best-known and understood examples include such things as regular backups (which may be full system backups or various types of incrementals) and database or directory replication, all of which typically involve large file or data transfers across the network. The key to keeping automated file transfers working properly is to add security envelopes (such as IPsec, SSH tunnels, or other forms of encryption/decryption technology) around those file transfers that cannot simply be replaced with more secure forms of file transfer such as SFTP.

Keyed Logins for Transfer (No Password)

Keyed logins are ideal for several reasons, yet face many of the same security implications as do typical passwords or passphrases. Keys are better than passwords or passphrases because key strength in both symmetric and asymmetric forms is many times stronger than passwords or passphrases. Whereas a password or passphrase seldom exceeds 128-bits or 16-bytes, as with a 16-letter password, secure storage and access mechanisms make it easy to support key lengths from 1024 bits through 4096 bits (if not larger) for short transfers (shared secrets, certificates, authentication data) where security concerns outweigh computational overhead. In turn, this permits regular exchange of message digest hashes and other key mechanisms to encrypt streams of data over the short term between communications partners so that compromising any single portion of the stream (theoretically possible, but unlikely in the timeframe between key changes) will still not compromise overall communications confidentiality and integrity.

Symmetric cryptography uses the same key to encrypt and decrypt data, whereas asymmetric cryptography uses a combination of public keys for encryption and private keys for decryption. Both are equally secure but asymmetric methods require more effort in practice. Keys need to be digitally signed and exchanged through a verified third-party entity or other neutral medium:

- SSH keys
- SSH keys with passphrase
- RSA SecurID cards

Keys can also be in the form of digital key fobs such as RSA Security's SecurID mechanism for two-factor authentication. These user-assigned hardware tokens generate authentication codes in periodic intervals using a factory-encoded random key, known as the seed. Each seed is different among SecurID devices and is typically 128-bits long.

Key-agreement protocols are effective where two parties agree on a key and both are influential to the outcome. The first and perhaps foremost public protocol established for this trend is the Diffie-Hellman exponential key exchange, where two parties jointly agree upon a generator composed using random numbers. In itself, this specifies no prior agreement or subsequent authentication between participants; it is therefore considered an anonymous key agreement protocol and remains susceptible to Man in the Middle (MitM) attacks.


Public key exchanges defeat such attempts using digitally signed keys, the very integrity and identity of which is sealed and delivered by a trusted third party and optionally signed by a CA. This mechanism is widely used on the Internet for SSL/TLS-based HTTP and FTP transactions.

Keyed Logins for Transfer (With Password)

Although it is entirely possible to password-protect a key, it may not be entirely useful for every situation. In this case, the password is usually an actual phrase and is therefore called a passphrase.

Passphrases have their arguable advantages and disadvantages. An ideal passphrase is generally better than a password because it is more than just a single word or word combination. It is an entire phrase of words, preferably of intermixed case and alphanumeric flavor, perhaps including symbols and non-printable characters for good measure. In fact, a passphrase can be much stronger (and often more difficult to recall) than a simple password. However, a passphrase that can be figured out with minimal computational effort (or by simple guessing) is no better than a poorly chosen password. You can have your public/private asymmetric keypair optionally secured with a password or passphrase, but such a step is futile if the passphrase is weak.

If this is so, why would you want to secure a key with a passphrase? One compelling reason is that, should the private key become disclosed, a proper password will keep it from ever being effectively used. Perhaps the key is kept on a USB thumb drive that is occasionally absentmindedly left in an unattended computer. In such a case, password or passphrase protection offers an additional layer of protection.

 Password selection is explained in further detail in the next section.

Securing Automated Transfers

Automation of file transfers is best facilitated by a form of public key exchange either by agreed Diffie-Hellman keys or Diffie-Hellman public key exchanges. Other key distribution systems of suitable strength and capability (RSA, DSA, ECC, El Gamal, and so on) are also worth consideration, but Diffie-Hellman is an established standard and primary example for this purpose.

Use asymmetric keys for protection and distribution of encryption keys and symmetric key algorithms for bulk automated transfers. This marriage of both technologies can better suit a variety of situations. Asymmetric keys can safeguard the symmetric keys from being purloined by unauthorized parties.

Preferred Authentication Methods

An ideal authentication method is one that gives strong server authentication and convenient user authentication. In practice, it may not happen this way; preferred authentication methods may sometimes appear architected to stymie most forms of end-user convenience. For instance, users do not generally consider it convenient to generate strong password choices more than 8 to 12 characters in length that include mixed-case letters along with alphanumeric or symbolic combinations (nevertheless, they are better off using such passwords, and should be well-informed as to the benefits and value that they deliver; it's often a good idea to offer access to a random password generator for those willing to use one who are also willing to observe the obligatory cautions against writing them down on a post-it note attached to their monitors). Realistically, the preferred methods are those that match many criteria, some of which are site- or installation-specific such as Microsoft's Encrypted Authentication or Extensible Authentication Protocol.

Even subtle differences to compile-time parameters or run-time configurations for simple SSH client and server combinations yield enough variance to warrant preferential treatment for some options and not for others. In one such case, the SFTP subsystem is largely built upon SSH protocol version 2 and is preferred over the many exposures and vulnerabilities in protocol version 1.

Perhaps the most obvious answer is to say, *stick with the standards*. Standardized, formalized, and ratified (in the case of IETF and RFC-related material) protocols, methods, and implementations are by far the best way to go. A preferred method should be mechanism- and implementation-independent and capable of interoperability between any number of platforms and applications. Such is the case for SSH, SSL/TLS, Diffie-Hellman, and other such security protocols. It should also be certifiably strong, validated to meet administrative and/or governmental criteria for secure data transmission or user authentication, ensure the high standards of privacy and confidentiality, and be relatively simple to use and deploy in any normal capacity.

Using Public-Key Mechanisms (With and Without Certificates)

PKI consists of applications, formats, procedures, protocols, policies, and public key cryptography mechanisms jointly working to ensure predictable, reliable, and secure communication channels. PKI establishes a level of trust within a potentially hostile and untrustworthy environment by providing for trusted third-party vetting of and vouching for user identities and binding of public keys to public entities.

This framework is established to provide authentication, confidentiality, non-repudiation, and integrity for messages in an exchange and is a hybrid system consisting of both symmetric and asymmetric key algorithms and methods. Like IPsec, PKI is only a framework and it also lacks specification of protocols and algorithms for secure exchange. As a framework, it consists of many separate useful components that include CAs, registration authorities, certificates, keys, and end users.

Public key cryptography is not the same as PKI; it's simply another name for asymmetric encryption algorithms, where PKI is an entire infrastructure. Public key cryptography can be viewed as a component of PKI, but they remain separate elements. Other components of this infrastructure or framework help to ensure user identities through certificate signing and the Diffie-Hellman exchange protocol for the negotiation of key exchanges. These pieces identify users, create and distribute certificates and encryption keys, maintain and revoke these certificates, and orchestrate the interoperation of all these implementation goals.

Where PKI will provide the means for handling certificates during a public exchange, basic public key cryptography does not make such allowances. Both methods are secure to use on an everyday basis, but the proper application of each method will be specific to the particular situation and any security considerations at hand.

Avoid Embedding Passwords in Scripts or Communications

Occasionally laziness and/or errant thinking will cause an administrator to make mistakes such as providing passwords for accounts in scripts or text files marked as readable by others. End users regularly put such critical information on sticky notes littering their cubicles; whether or not such exposure is the fault of the administration staff, it remains their responsibility.

Please resist the urge to embed any password within any script or leave it publicly readable in any case, even if you can cleverly disguise its meaning, obfuscate its form, or prohibit its accessibility. Although it may briefly alleviate administrative strains, such as reminding a user what their password is for the umpteenth time, it will increase administrative discomfort when someone else discovers this crucial piece of information and takes advantage of it in some unauthorized manner.

Avoid Weak Passwords

There are many instances where you must issue passwords for authentication purposes. Given all other security precautions for set up, management, and tear-down of cryptographically secured communications, strong password choices must still be made to ensure the confidentiality of each end-user transmission.

Suppose a public HTTP or FTP server provides basic login features for your clients, and it is deemed impractical to issue proper public/private keypairs for each one. In this case, you should take several extra proactive precautions against poor password selection.

Typical usage scenarios involve a fixed-length password of mixed-case alphanumeric passwords. For example, it is common to see password fields restricted to a minimum of 8 to 12 mixed characters. The mixed-case component becomes apparent with case-sensitive systems where UNIX and Unix represent two different forms of the same word. This case-sensitivity and mixture with digits and symbols greatly increases the variability of the password by offering extended combinations and permutations. This decreases susceptibility to brute-force dictionary attacks, in which trial-and-error sessions expose easily guessable password choices.

Also, many modern login systems—beyond interactive prompts—provide some means of cross-checking the end user's password choice. Items deemed unsuitable for general-purpose or production use are issued warnings, and the user is commanded to make a stronger choice. Such systems enforce correct behavior for every password whether being newly created or changed from an existing one.

The following list highlights a few pointers for strong password selections:

- Mix numbers and letters; mix case; include punctuation and symbols for added strength
- Merge one or more words side-by-side or interleaved; in the second example, two words UPPER and lower might become UIPoPwEeRr
- Combine a root word with an appendage: the root word need not be a dictionary term, but it should be pronounceable; the appendage is either a prefix or suffix, such as prefort or forttable, where pre is a prefix and able a suffix attached to the same root word, fort
- Choose an arbitrary phrase and select specific letters from each word; for example, mess with the bull, get the horns becomes mWTbGth
- Find something memorable but nothing too personal that someone else could easily discover, such as significant dates, personally identifiable information, pet names, and so on
- Never use the same password for multiple logins; once an unauthorized party discovers this universal password, guessing the login or account name is a simple hit-or-miss affair at popular online retailers, payment systems, and online banks.

The most important thing to remember is that even the strongest fortifications can be infiltrated where the keys are effortlessly guessed or reproduced. For all the ironclad security properties provided by the best cryptographic algorithms, implementations, and frameworks, each one is only as strong as its weakest key—or in this case, password or passphrase.

Summary

This chapter explored the cryptographic protocols and frameworks necessary to construct reliably secure communications channels. IPsec, SSL/TLS, and SSH are just a few of the standardized solutions to meet the need for secure communications. Some of these solutions represent only part of the answer because they require other components to function in any capacity (for example, IPsec and SSL/TLS versus SSH). Some of these solutions may integrate and work with others or present an alternative choice to another (for example, IPsec versus SSL/TLS VPN). The SSH protocol covers secure communications as well as secure file transfer and will work over an IPSEC or SSL/TLS link.

The chapter also discussed that although file transfer itself is closely associated with the file transfer protocol, not every file transfer is facilitated by FTP. Many secure file exchanges occur over HTTP and this is another area where SSL/TLS is commonly applied. Files may be transferred over a number of technologies, some of which may or may not inherently support SSL/TLS, IPsec, or other cryptographic components that make for secure transfers. Tunneling and VPN technologies become an attractive option for an all-encompassing solution to secure legacy applications or those that do not directly work with typical cryptosystems and cryptographic communications protocols.

Finally, the chapter talked about the strengths of public key-based cryptosystems including:

- Proper key exchange is a vital component in trusted party-to-party communications across shared channels
- Diffie-Hellman first defined the public/private keypair concept, but many other worthy key algorithms like RSA, DSA, ECC and El Gamal exist
- The strength of a cryptosystem comes from the algorithm, initialization vectors, and the length and secrecy of the key

The next chapter further explores the subjects of SFTP, FTPS, and IPsec with more attention to comparison and contrast between these technologies.

Chapter 4: Compare/Contrast SFTP, FTPS, and IPsec

In previous chapters of this book, we've examined the issues involved in finding and replacing or strengthening the types of file transfers used for so many important tasks in so many enterprise settings. Along the way, we've dug into various key tools and technologies that may be used to help this effort along. Key elements in this laundry list of possibilities have included the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) as well as IP Security (IPSec) and the ever-popular Secure Shell (SSH, which today includes both SSH-1 and SSH-2 versions). We've also explained how these various elements may be employed to help boost security for file transfers and to assure reasonable levels of security, confidentiality, and control over file transfer activities and the content and control information they typically convey.

In some cases, it may make sense to simply wrap existing implementations—such as scripts; legacy tools; or other file transfer code for which there may be no source code available nor an easy way to remove older, less secure file transfer applications such as FTP—and to replace them with more secure alternatives such as SFTP or SCP. These are cases in which adding an extra security wrapper, such as IPSec or SSL/TLS can provide the kinds of security, confidentiality, and control that the file transfer programs (and other code inside the wrapper) may lack.

In fact, a Virtual Private Network (VPN) comes very close to realizing the old puzzler about how to have one's cake and eat it too. That's because proper use of VPN technology permits public networks—most notably, the Internet—to serve admirably and securely for ferrying private, encrypted communications between senders and receivers. In fact, the more the VPN software can direct traffic through its transports, the more secure ordinary network communications become, especially those that may not be very secure (or secure at all, such as FTP, Telnet, and unencrypted email clients).



The connection medium used to distribute information is a major determining factor when choosing a VPN solution. Part of this choice is platform-specific and another part is protocol-specific.

The type of VPN arrangement required between each participating endpoint is another major factor to consider when deciding on a solution. There are connection endpoints, or those that establish actual VPN connections from one network to another, and data endpoints that send and receive private information across VPN links. In some situations, these two entities may be part of the same unit or spread across several separate units.

Host-to-Host VPNs

Host-to-host VPN is designed to replace costlier frame-relay and leased-line arrangements that might otherwise be used to secure communications between the host's business partners, subsidiaries, subscribers, or vendors. This configuration provides end-to-end security as the VPN software is installed on each host computer that runs whatever applications that need more security, including file transfer.

Host-to-Gateway VPNs

In a host-to-gateway arrangement, the VPN protects data in transit between a host on some local network segment and a remote gateway device—but not data on the remote network. This may also be called a remote access VPN. However, please note that this configuration only secures part of the communications link and does not provide complete, end-to-end security (though ideally it secures all public links across the Internet, which are presumably in greatest need of security and enhanced protection).

Gateway-to-Host VPNs

In a gateway-to-host VPN, one endpoint consists of a network serviced by an intermediary VPN gateway endpoint that communicates with a single remote network contact over an intervening network. Data moves between gateway and host endpoints under the protection of VPN-supplied encryption but traverses the local network segment behind the gateway unmodified. Likewise, this configuration only secures part of the communications link and does not provide complete, end-to-end security.

Gateway-to-Gateway VPNs

Gateway-to-gateway implies that both connection and data endpoints differ in their VPN topologies. Traffic is protected between connection endpoints but travels in native form in either direction between data endpoints. This may also be referred to as a site-to-site VPN, or mutual interconnection between entire networks.

As a usage scenario, consider an organization comprised of a single central office with several branch offices, all of which maintain their own intranets. VPN protection is mandated for safe passage of information across the Internet but not within each respective intranet. A series of border servers acting as VPN gateways can exchange information securely and freely with every other recognized and valid intranet through the intervening network medium (the Internet).

Software Solutions

Software-based VPNs require a computer capable of handling intensive encryption and compression routines above and beyond whatever existing desktop or server applications may already be in use, with minimal impacts on system performance and application responsiveness.

Security becomes more challenging as the number of VPN installations increases. Likewise, as the VPN orientation focuses on either internal or external communications if not both ways, complexity likewise increases. When establishing matched system architectures, similarity among load-balanced systems must be carried to the logical extreme. Each platform must use identical hardware, and software images must be compatible down to the patch-level (for cloned or clustered servers) and for all access permissions. As the old saying goes, one rotten apple can spoil the whole barrel: the same thing is true for each and every system in a load-balanced group—namely, that mismatched elements can cause instability problems, severely degrade performance, or cause load balancing to fail altogether.

Each installation must be considered at risk in varying degrees according to its orientation toward internal and external sources. The security of any VPN solution is directly impacted by the supporting operating system (OS) environment, which must therefore be taken into consideration for each and every installation. This issue affects both client and server roles in any VPN arrangement equally.

Hardware Solutions

Certain intermediate network appliances are specifically designed to offload VPN processing from host machines. Such appliances may be placed inline on the network, and provide a viable and workable VPN solution particularly for medium- to large-scale network environments. VPN appliances dutifully handle encryption, compression, and other VPN-related tasks, thus accelerating the performance of VPN-based network transactions. One caveat that applies to vendor-specific VPN solutions, whether hardware or software oriented, is that one vendor's IPsec implementation may not always work with another's. There are VPN routers and appliances on the market that may be compatible only with identical makes and models. Thus, you must match these devices on either side of the network connection that they straddle. You'll want to investigate detailed product specifications and disclaimers carefully, and make your selections accordingly. Where homogeneous components may be neither feasible nor desirable, you will need to test candidate configurations carefully to make sure things work properly before deploying any equipment into the field. Essentially, this means setting up a test lab that resembles the planned deployment environment as much as is technically possible, and subjecting it to usage scenarios and workloads that also match the target networks. Only those solutions that actually work, and work sufficiently well to meet user requirements, should ever reach users' hands.



It is worth noting that most hardware solutions, if any, also do not provide complete, end-to-end encryption.

Intranet Solutions


Business assets may be (and usually are) dispersed over multiple, disjointed regional territories. Nevertheless, each location can use dedicated equipment to establish VPN partnerships between pairs of sites (or between spoke ends and hubs, in the hub-and-spoke model). Branch and central offices can be mutually conjoined across the Internet by using a VPN solution.

This approach offers significant cost savings as compared with pricier frame-relay or leased-line arrangements (as explained earlier, the savings can be substantial, even when additional services are included in the mix, often on the order of thousands to tens of thousands of dollars per month for medium- and enterprise-class organizations). However, please bear in mind that gateway solutions may not provide you with the level of security you need. Remember that to be 100% protected, you must ensure that your VPN solution provides complete, end-to-end (application-to-application) security.

Extranet Solutions

Business alliances may encompass two or more companies closely cooperating with intranet-based VPNs to coordinate and provide specialized services to customers and clients. This affords such companies the ability to interoperate within a mutually shared and secured environment via the Internet to enhance business operations and provide applications and services that take advantage of their combined information and processing assets.

An extranet solution is probably best characterized as providing additional Layer-3 security to extend wide area networks (WANs) to other regional territories or business partnerships. Because the sum of the security risks across multiple organizations is greater than the individual risks that each participant must shoulder, owing to additional exposures and vulnerabilities that appear “in the cracks” between organizational boundaries and controls, this helps to explain heightened needs for authentication, access controls, security, and auditing that are typical for extranets—and in the VPN environments that support them.

 For more information about Layer-3 security in VPNs, see the Cisco white paper “[Beyond the Basics: Technologies for Compelling Layer 3 VPN Services](#).” Tim Greene’s Network World story “[Layer 2 vs. Layer 3 VPNs](#)” zeroes in on issues specific to MPLS in this regard, and may also be of interest.

Tunneling Protocol Options

As described in the preceding chapters, many types of protocols may be used to create secure tunnels between pairs of endpoints. That said, we only explore IPsec, SSL/TLS, and SSH here, as we discuss their respective advantages and disadvantages as they relate to securing automated file transfers

Tunneling requires a carrier protocol, an encapsulating protocol, and a passenger protocol. The carrier protocol is used by the network upon which information travels, and the encapsulating protocol envelops the passenger protocol, which includes all data being transported.

It is important to understand the primary distinction between an IP-layer tunneling mechanism such as IPsec and a higher-level counterpart such as SSL/TLS or SSH: high-level services protect a single protocol and low-level services protect a single link.

In the case of SSL/TLS and SSH, these higher-level services safeguard individual protocols such as FTP, Telnet, SMTP, and so forth, most typically on a one-at-a-time basis. For a lower-level framework, such as IPsec, anything may traverse the link it envelops—even protocols that may not be routable can be encapsulated easily and transparently for delivery across the tunnel.

Advantages

There are numerous protocols used for tunneling which may be used to create VPNs. As this guide largely concerns the distinctions between IPsec, SSH, and SSL/TLS solutions, only those options are explored.

Advantages of IPsec

IPsec has its relative strengths and merits, particularly in the enterprise computing environment. Chief among these are the following:

- Protects any protocol—IPsec uses IP for multi-protocol encapsulation
- Protects any medium—IPsec utilizes multi-protocol encapsulation over any single-protocol medium
- Total transparency—IPsec background services have no visible impact on end users
- Total scalability—IPsec can be applied to different networks of various sizes and capacities
- Total independence—IPsec will route any number of network-based applications
- Connection-class controls—IPsec supports network-based authentication
- Economy of bandwidth—IPsec supports compression for improved throughput
- Protocol bridging—IPsec provides multi-protocol local networks across a single-protocol backbone
- Connectivity solutions—IPsec provides workarounds for networks restricted by limited hop counts such as Appletalk or NetBIOS
- Total network continuity—IPsec interconnects discontinuous subnetworks
- Extended coverage—IPsec allows VPNs to operate across WANs

Advantages of SSL

SSL also has numerous interesting noteworthy aspects, as the following list highlights:

- Completely universal—SSL's all-purpose applicability instantly creates a VPN client using any modern Web browser (older versions may not support SSL, so be sure to test older browsers that may be employed in your user population, knowing that VPN requirements may force upgrades to occur)
- Total flexibility—SSL supports additional applications without firewall configuration changes
- Total transparency—SSL background services have no visible impact on end users
- Cost effectiveness—Existing SSL implementations conserve both time and budget
- Unrestricted movement—Many SSL VPNs use the SHTTP port, allowed to pass through most firewalls

Advantages of SSH

As a protocol originally developed to replace remote logins, terminal emulation, and remote login, SSH excels at enabling users to get things done via secure remote access. Enhancements added since its inception in 1995 enable this protocol to support tunneling and port and X11 connection forwarding, and sets up a secure, encrypted channel between local and remote hosts, with strong authentication from user to remote host and vice-versa. SSH protects message content and ensures message integrity via encryption for confidentiality and message authentication codes for integrity. Other advantages include the following:

- Easy to deploy and maintain
- Virtually all computing platforms are supported from Windows to mainframes, where nearly all of these versions interoperate very well
- Provides true end-to-end security for the applications it protects
- Provides supports for strong two-factor (or multi-factor) user authentication
- Provides authentication for computing hardware (for example, servers and hosts) as well as users
- Commercial implementations offer advanced features including tools to automate replacement of scripted file transfers with secure alternatives
- Port forwarding may be used to transparently secure file transfers, when replacing insecure FTP with secure alternatives (SFTP, SCP) isn't an option
- Easy to secure HTTP-based or homegrown file transfer applications without altering programs already in use
- Works with IPsec and SSL VPNs without requiring substantial alterations on either side of that interaction
- Can be used to protect virtually any application, so once you have decided on SSH, you have a protocol that can solve many security issues at the same time; it not only enables easy replacement of FTP but also helps to solve the Telnet problem

Disadvantages

There are also disadvantages to using either IPsec, SSH, or SSL/TLS for securing a network environment. As is the case with virtually any security solution, measurable benefits relate to how well and completely any product addresses security concerns and conditions.

Disadvantages of IPsec

Among other implementation-specific challenges, IPsec is subject to the following distinct disadvantages:

- Various implementations are not necessarily compatible, especially when using out-of-the-box defaults, and can require significant testing and deployment time
- IPsec is only as secure as the gateways through which it passes—vulnerable intermediate devices can undermine trust
- Not secure end-to-end—IPsec secures two endpoints, not applications and users
- Endpoint-oriented—For IPsec, strong authentication functionality operates without user IDs
- Not fully featured—IPsec lacks digital signatures and public key functionality
- Not unobservable—IPsec is not designed to defend against traffic and protocol analysis
- Intrusive—For IPsec, OS integration at the network level is required for it to be usable
- Expansive—For IPsec, encryption of small packets produces large overhead to payload ratios

Likewise, SSL presents its own unique challenges and considerations when it comes to solving the security equation:

Disadvantages of SSL

SSL requires much more effort on the part of the end users to establish an ad hoc trust relationship for the purpose of exchanging messages, information, or data. Although this is more convenient from an administrative standpoint, it's also an inconvenience in that typical network-level authentication mechanisms do not directly apply to such traffic (as is the case for IPsec).


The following list highlights additional disadvantages of SSL:

- Inherently insecure—Each SSL implementation is only as secure as the local computer on which it runs
- Inherently incomplete—For SSL, strong authentication is required for enterprise-grade VPNs
- Untrusted sources—For SSL, administrators cannot specify privileges using client origin IP addresses
- SSL VPN solutions secure Web-based applications by default, but additional software must be installed on each PC to provide more advanced functionality
- Rarely provides end-to-end protection as SSL VPNs normally use a client-gateway architecture

Disadvantages of SSH

SSH was not originally designed as a VPN, though it works to deliver much of the same functionality that VPNs offer, most importantly, a secure connection. SSH is a software product and as such needs to be installed and configured. Other disadvantages of SSH include:

- The most secure and advanced implementations of SSH are commercial, especially where automation tools and infrastructure extensions may be needed or desirable (for example, for automating switchover from insecure file transfer applications to more secure alternatives)

 SFTP and SCP are part of SSH and are included as an integral part of commercial products.

- SSH is not integrated at the OS level, so software will have to be installed on most platforms

In actuality, SSL/TLS and IPsec solutions can coexist quite happily in secure network environments and represent cogs in a much larger-scale security infrastructure. SSH often serves as an alternative or replacement for either or both schemes, because it is simpler, less costly, and less labor-intensive to implement. That said, SSH also coexists quite successfully with IPsec and SSL/TLS technologies.

Protocol	Pros	Cons
IPSec	Protects individual links	Costly, difficult to implement
	Encapsulates all kinds of protocols	Differing vendor defaults cause problems
	Protects any protocol	Gateways provide points of exposure
	Protects any medium	Secures endpoints, not applications or users
	Transparent to end users	Authentication works without user IDs
	Highly scalable	Lacks digital signatures, full PKI support
	Application independent	Does not defeat traffic/protocol analysis
	Works with network-based authentication	OS must be integrated at network level
	Supports compression	IPSec VPNs are normally deployed in gateway-gateway configurations and do not provide full end-to-end security
	Interconnects discontinuous subnets	All implementations are not NAT friendly
Enables VPNs to operate across WANs	Heavy overhead encrypting small packets	
SSL/TLS	Works with modern Web browsers	Only as secure as computers it runs on
		Does not necessarily provide end-to-end support for legacy applications such as FTP
		Does not provide support for non-Web applications without installation of additional software.
		Often reliant on OpenSSL libraries
	Add applications without firewall changes	Lacks built-in strong authentication
	Background services don't impact users	No privileges associated with client IP
	SSL implementations save time and money	More effort to establish trust relationship
	Use SHHTTP port to bypass firewalls	Network-level authentication does not apply

SSH	Supports various types of remote login	Not originally designed for VPN use
	Support various forms for authentication, including password, public/private keys, certificates, RSA Secure ID, Kerberos, GSSAPI, PKI, and so on	Must be integrated with authentication
	Supports tunneling	
	Forwards X11 connections & TCP ports	Most secured implementations are proprietary
	Uses SFTP or SCP for secure transfer	Some expense, upkeep involved
	Confidentiality and integrity built-in	Apps may be needed to secure transfer/copy
	Easy to deploy and maintain	Not integrated at OS level
	Easy automating secure transfers, scripts	
	Port forwarding transparently secures xfer	
	Easy to transparently secure HTTP apps	
	Works with IPsec VPNs	
	Works with SSL/TLS VPNs	
	Delivers true end-to-end security	


Table 4.1: Summary of tunneling protocol pros and cons.

Security Considerations

There are many elements to consider when establishing or enforcing an overall security scheme. Policies, procedures, and people's roles all must be defined so as to operate in some particular capacity, with some set of restrictions, and under some formal method to ensure proper compliance and regulatory controls are in place. Each element must be broken down into its basic parts and defined according to the roles it fills. Every component must be scrutinized and defined according to its terminology, rules, and processes.


Strong Encryption and Authentication Using Standard Algorithms


The use of strong encryption provides additional security against skilled and brute-force attempts to gain access to securely encrypted data. The use of standard algorithms ensures a reasonable level of resilience to such attacks when applying appropriate keys or passwords of appropriate bit strength. Implementing strong access controls provides the authentication necessary to safeguard against and account for potentially hostile user activities.

 Standard encryption algorithms are published and knowledge about their operation is widespread. Most security experts believe that such openness leads to widespread efforts in the security community to expose and deal with weaknesses, vulnerabilities, and exposures, and rapid abandonment of algorithms deemed not worth saving, repairing, or maintaining. Proprietary or closed encryption algorithms often turn out to be weaker, and the results of their compromise more severe, because the lack of openness does not foster common understanding of and appreciation for their strengths and weaknesses. Examples of standard security algorithms include DES, triple DES, RC4 and RC5, and the Advanced Encryption Standard (AES).

Both encryption and authentication play a critical role in a much larger security picture: together, they establish a firm foundation for trust between servers, services, and subjects. As such, they should use only established, standard algorithms to ensure reliable and safe operation. As a rule, standard algorithms must withstand lengthy and thorough public reviews, private audits, and widespread inclusion with other security applications and services, which has the added benefit of ensuring interoperability among multiple platforms and architectures.

Buying into a non-standardized implementation potentially ties an organization into a vendor-specific viewpoint that may not scale well or adapt to changing conditions such as business mergers, acquisitions, or newly formed partnerships. Where possible, buyers should seek out certified solutions for deployment (in the US, this would mean preferential selection of implementations certified to comply with FIPS 140-2, for example).

 For various lists of products validated to conform to FIP 140-2, please visit <http://csrs.nist.gov/cryptval/vallists.htm> (especially the FIPS 140-1 and FIPS 140-2 Vendor List).

 In his classic security book *Secrets & Lies: Digital Security in a Networked World*, security maven Bruce Schneier makes the point that “if a [security] primitive is only secure if it remains secret, then it will only be secure until someone reverse engineers and publishes it.” He goes on to point out that proprietary security primitives that have been so outed include digital cellular communications encryption, DVD encryption, FireWire encryption, various Microsoft encryption algorithms, various smart card electronic commerce protocols, the secret hash function in SecureID cards, and even the protocol that protects Motorola’s mobile MDC-4800 Police Data Terminal. His most telling point is that “public primitives are designed to be secure even though they are public; that’s how they’re made. So there’s no risk in making them public.”

Consider How Control and Data Channels May Be Handled

Keep in mind that a solution such as FTP via SSL/TLS is a patchwork fix to a much more complex problem than it can address. Remember that with FTPS, the control connection is always encrypted but the data channel may not be likewise secured (in other words, account and password information will be protected but the contents of files transferred may be sent in the clear, which provides some improvement over plain-vanilla FTP, but still exposes files to snoopers). Because the authentication process must be fully protected at all times, it should support full-time use of encryption to thwart eavesdropping attacks against the system by intermediary devices or parties. That said, pre- and post-process encryption files do affect how they will be handled in transit. In the first case, a pre-processed encrypted file passes through the data channel unmodified; in the latter case, a file must be processed prior to its transmission across the network medium and incurs a performance penalty.

Security Policy and Compliance Requirements

The goal of any security or privacy policy is simple: to help employees *do the right thing*. All security and privacy policies must take into account regulatory requirements during the drafting, designing, or updating process. Not only must such access and administrative controls already exist for many organizations, they must also be demonstrably compliant with HIPAA, SOX, or whatever legislation, regulations, or contractual agreements with third parties that may apply to the organizations in question.

When to Use SFTP

SFTP is built upon SSH and provides an additional secure encapsulation of file transfer command and data streams. SFTP uses the existing SSH protocol to create and manage multiple command and data streams to safeguard against eavesdropping by intermediate parties that may attempt to harvest sensitive information, including login credentials and files.

SFTP may be used anywhere SSH is implemented in client or server form. Both console and graphical clients are available for most OSs, which makes SFTP an excellent fit for most enterprises and inter-organizational arrangements. Use SFTP readily wherever plain-vanilla FTP installations need a security upgrade. This means you should deploy SFTP where personal identity information or business-critical confidential data may be acquired, retained, or processed, as when handling medical or financial records or when transferring proprietary or sensitive data among business partners (or between sites inside some organizations). In fact, it's a prudent practice to consider SFTP first before turning to stopgap measures such as FTP over SSL/TLS solutions because SFTP offers better security and is often easier to manage than such other alternatives.

It is interesting to note that many organizations now use SFTP exclusively across their enterprises to secure all data in transit once and for all. Overall security risks may be reduced significantly when using software such as SSH Tectia on each computer in the network, and it is a cost-effective way to secure just about any network. One conclusion that may be drawn from these observations is that SFTP should be used whenever an organization seeks to secure its data with no (or in the worst case, only minimal) changes to its applications or infrastructure.

When to Use IPsec

IPsec is best deployed in situations in which complete access to hardware and local software, including modification of the OS and related administrative utilities, is permitted. Network-based authentication makes IPsec most suitable for connection-oriented access and authorization controls but a poor choice for client-oriented permission granularity. Thus, IPsec might be best suited for securing conferencing or chat-type interactive applications, but some other approach that integrates access control lists (ACLs) or role-based security might work better for providing remote access to file stores or document repositories.


That said, IPsec should invariably be used in concert with other site-specific security measures. Firewall configurations, intrusion detection systems (IDSs), and possibly user-held identity tokens or biometric identification devices may also be necessary to establish the right levels of enterprise security. Likewise, the processing demands that large user communities can impose on network access and transfers may also mandate deployment of intermediate network appliances as well to handle encryption and decryption, help with authentication and identity management, serve as remote access gatekeepers, provide VPN support, and so forth.

When to Use FTPS

FTPS involves the use of a separately created secure channel with an existing, full-fledged FTP server by adding another, secure file transport layer based on SSL/TLS. Essentially, this method adds SSL-capable send-and-receive functionality to existing FTP protocols and server applications. However, this approach only partially addresses FTP security issues, which lacks other strong components that are routinely included as part of an SSH/SFTP subsystem. Strong authentication and user-based transaction accounting are two primary examples of what else comes as part and parcel of such offerings, and helps explain why these kinds of solutions have proved so popular in enterprises and organizations of all sizes and scales.

Security

Standard SSH provides a remote login shell for the user. It is possible to restrict how this shell behaves for those clients—that is, to more precisely specify what types of access and actions will be permitted and which ones may be denied. Setting up a more customized shell environment for clients may require more complex SSH server configurations. FTPS, however, offers no ability to execute arbitrary commands outside those built-in to the FTP command set. Here again, the neatly compartmentalized implementation that FTPS delivers helps to explain its widespread popularity and use.

 For information about how to restrict SSH access, see <http://www.snailbook.com/faq/restricted-scp.auto.html>.

Flexibility

Because FTPS is a straightforward extension of the existing FTP infrastructure, it is supported in many proprietary and open source server applications. In fact, enabling FTPS in most of these services usually involves setting only a simple configuration parameter. No additional firewall configuration or supportive services are needed.

Certificates and Certificate Authorities

Where SFTP uses keys or certificates (among other mechanisms), FTPS uses certificates. FTPS does not support the chains of trust paradigm facilitated through Certificate Authorities (CAs), nor does it require that parties first exchange confidential security details before forming a trust relationship. That said, SSH supports a wide range of authentication mechanisms, including client keys (which must be maintained on the server) and certificates issued by CAs both public and private.

The Demilitarized Zone

In relation to computer security, a Demilitarized Zone (DMZ) refers to a network segment that resides between an internal and external network, such as when an FTP, Web, or database server is located just outside the intranet but between the internal intranet and the external Internet. Typically, a DMZ contains devices that are Internet accessible on a specifically designated subnet that is isolated from protected internal resources. There may be one or more DMZs, separated by firewalls, each housing separate servers running different services; for example, one DMZ for HTTP servers and another for public DNS and SMTP.

FTP Placement Inside the Firewall

The firewall plays an integral role in monitoring and controlling network traffic that passes in and out of any organization's perimeter. A typical one-tiered configuration places LAN endpoints (clients and servers) in close proximity to the network, separated perhaps only by a firewall or perhaps a router. Administrators can log in and supervise FTP behavior easily by placing the server inside the firewall. Traffic flow between DMZ segments, LAN endpoints, and Internet-reachable destinations can be captured, observed, and adjusted quickly as needed.

A two-tiered architecture includes front-end client-oriented servers and back-end server-oriented servers, perhaps contained within a single DMZ or perhaps separated by an internal firewall or router with active ACLs. This arrangement is suitable for most business networking environments although banking, financial, and health institutions may require more robust security configurations, as described further in the following paragraph.

Three-tiered architecture begins with a front-line assembly of client-oriented service application servers and middle-man application servers that negotiate transactions between front-end servers and better protected back-end servers. These layers may be called the presentation, business, and back-end tiers, respectively. Business and presentation tiers reside in a DMZ behind an Internet-facing firewall, with discrete firewalls separating them from the back-end tier. This arrangement provides more granular control over network-based application transactions than with one- or two-tiered configurations, where information can be checked, controlled, and handled separately at each point in the transaction chain.

Choosing a File Transfer Solution

Ultimately, the file transfer solution you choose will be determined by your specific business needs and the kinds of connectivity that extranet and remote access requirements entail. Local compliance policies and security protocols will help to determine what will and won't work for particular network situations. Whatever those circumstances might be, however, the type and relative sensitivity of data being transferred must also play a key role in driving your selection process.

At-risk servers that house personal identity information (aka personally identifiable information) exert strong attraction on would-be data thieves and require special attention and extra protection. Likewise, sensitive or confidential business information demands additional encapsulation and protection. Prudence and regulation dictate that both types of data assets be afforded strong levels of authentication, access controls, and encryption when such data is stored and whenever it's transmitted to any authorized users. Thus, any file storage and transfer system that may be designated as at-risk or for which a high-threat level is assessed must use a secure file transfer product such as SFTP.

Total Cost of Ownership

Careful risk analysis demands that the cost of your security solution should not exceed the value of the protection it provides and the exposures it helps to avoid or mitigate. The effort involved in performing such risk analysis can exceed the amount of effort involved in the implementation of any security solution that results from such analysis, but the costs of such effort must be considered as you work toward an appropriate solution. The investment decision for endpoint security thus offsets the value of protection and risk avoidance against cost and must budget for the total costs of ownership (TCO) over the useful lifetime of whatever solutions may be included in your organization's security infrastructure.

Both in general and where file transfer solutions are concerned TCO encompasses many elements:

- Security software licensing
- Hardware costs
- Initial deployment labor
- Ongoing maintenance and administration costs
- End-user support and education costs

Despite the widespread use of antivirus and intrusion detection or intrusion prevention software at the enterprise level, viral outbreaks and security breaches do sometimes occur even in enterprise environments. As long as security measures are in place, there is a concomitant need to block any potential countermeasures. On the one hand, information theft occurs much less regularly than do malware-driven incidents (according to statistics from security information clearinghouse IT-Harvest). For example, malware incidents outnumber data theft incidents by at least four orders of magnitude. When you stop to consider the former includes all incidents related to viruses, spyware, and adware and the latter refers only to successful attempts to illegally access unauthorized information, the claim doesn't sound as outrageous as you might think. On the other hand, one single incident of data theft has the potential to cause thousands to millions of dollars of negative financial impact. By enforcing strict policy and regulatory compliance for connecting devices and taking other risk-avoidance, transfer, or mitigation steps as may prove necessary (such as establishing the level of loss an organization is willing to absorb, then obtaining extraordinary loss insurance), proper endpoint security products should provide reasonable protection against tools and automation software that enable attackers to exploit more vulnerable targets.


Cryptographic security is available in various packages whose pricing ranges from free, standardized modular components, to complete, commercially available turnkey solutions that suit a variety of business arrangements and needs. Also remember that when the time comes to decommission or transfer ownership of equipment to third parties, it is essential to cleanse site-specific information, including encryption keys or certificates, from each endpoint computer. Making these vital bits of information accessible to outsiders is like handing the keys to your kingdom to a total stranger.

Assessing Overall Applicability

A well-documented security policy lays the foundation for any secure infrastructure. Such a document, or collection of documents, conveys both intentions and decisions as to the roles that security plays within the organization. A security policy should identify critical and important business resources, activities, and operations. Smaller organizations may have a single security policy document that defines all necessary subjects and objects, where larger organizations may require an overarching general policy document with additional support from additional, specific policy addenda, such as policy documents that govern remote access, firewall behavior, acceptable use, and records and information privacy/confidentiality requirements. Resource usage constraints, remote administration and access controls, and information protection between internal and external sources must all be spelled out in such documentation. All of these topics must be tailored for individual sites, business units, and organizational functions to conform to existing organizational policies and current business needs.



The SANS Security Policy Project (<http://www.sans.org/resources/policies/>) includes a comprehensive set of tutorials, information, resources, and examples of a complete enterprise-level security policy document collection along with information about how to craft and maintain that collection as passing time and changing circumstances will demand.

 The National Institute of Standards and Technology (NIST) offers a collection of [Special Publications](#) that include numerous documents about security policy and related documents and procedures. You can also access two compendia of books on the topic of crafting security policy on SearchSecurity.com—namely “[Security Policy by Example](#)” and “[More Security Policy by Example.](#)”

Summary

Ultimately, securing automated (and other) file transfers within an organization requires finding and identifying the means whereby they occur, and either adding an additional security wrapper to boost security levels or replacing insecure file transfer tools with secure alternatives. Thus, when it comes to designing and implementing secure solutions for file transfer, responsible individuals in enterprises and organizations would do very well to heed the following recommendations:

- Encrypt all data between any two computers involved in file transfer (thereby also implementing end-to-end security, as will be automatically ensured by using SFTP).
- Always encrypt usernames and passwords and publish and enforce proper guidelines for password strength and complexity, providing user education to match.
- Seriously consider how many changes you can make to your file transfer infrastructure. If circumstances dictate little or no changes are possible, SSH is probably the best choice; only if more funds and time are available, should you contemplate implementing other technologies.
- Seriously consider how much time you can allocate to implementing changes to a file transfer infrastructure. IPSec deployments often involve major time and resource commitments, whereas SFTP projects may be undertaken quickly, and normally require only minimal resources.
- Decide how you may best authenticate file transfer processes. If you can leverage existing directory services, such as Microsoft Active Directory (AD), that may make sense. Otherwise, it may be necessary to acquire and implement a public key infrastructure along the lines of PKI.

Download Additional eBooks from Realtime Nexus!

Realtime Nexus—The Digital Library provides world-class expert resources that IT professionals depend on to learn about the newest technologies. If you found this eBook to be informative, we encourage you to download more of our industry-leading technology eBooks and video guides at Realtime Nexus. Please visit <http://nexus.realtimepublishers.com>.