


Realtime  
publishers

"Leading the Conversation"

# *The Shortcut Guide<sup>™</sup> 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 Realtimedpublishers.....	i
Chapter 1: File Transfer Security Issues.....	1
FTP: The 10,000 Foot View .....	2
How FTP Is Used.....	2
FTP Behavior .....	3
Where Is FTP Used?.....	7
What Makes FTP Insecure.....	8
Passwords and Contents in Plain Text.....	8
Lack of High-Level Integrity Checks .....	9
Separate Channels and Random Port Allocation.....	10
Abuse of Built-In Proxy Features .....	10
Other Insecure Transfer Methods and Models.....	10
Understanding the Role of Automation .....	12
Scheduled Tasks.....	12
Command-Line FTP and OS-Level Schedulers .....	13
Centralized Job Scheduling Solutions .....	13
Built-In Application Capabilities.....	13
Use of External Tools and APIs.....	13
Use of Network Drives .....	13
File Transfer and Regulatory Compliance.....	14
Privacy and Confidentiality Issues.....	14
Relation to SOX, HIPAA, PCI, and Other Compliance Regimes .....	15
Summary.....	16

## 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](mailto: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.


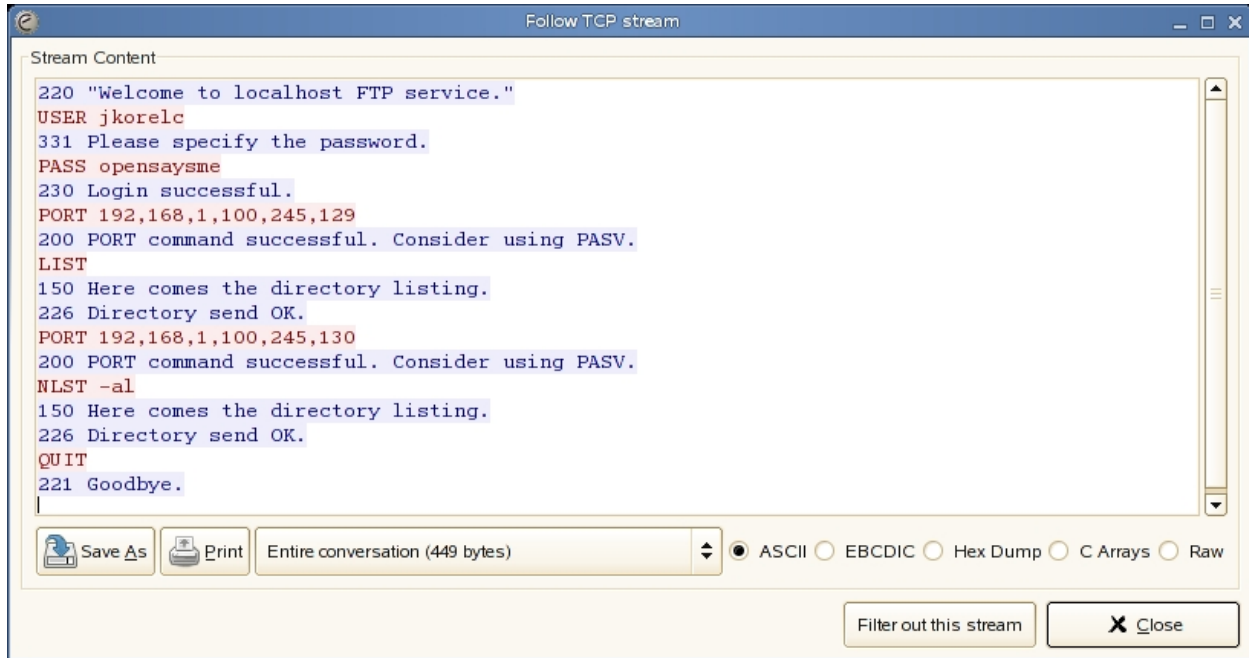
 In active mode, the client opens a random port (> 1023), sends the server the random port number on which it is listening over the control stream, and waits for a connection from the server. When the server initiates the data connection to the client, it binds the source port to port 20 on the server.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



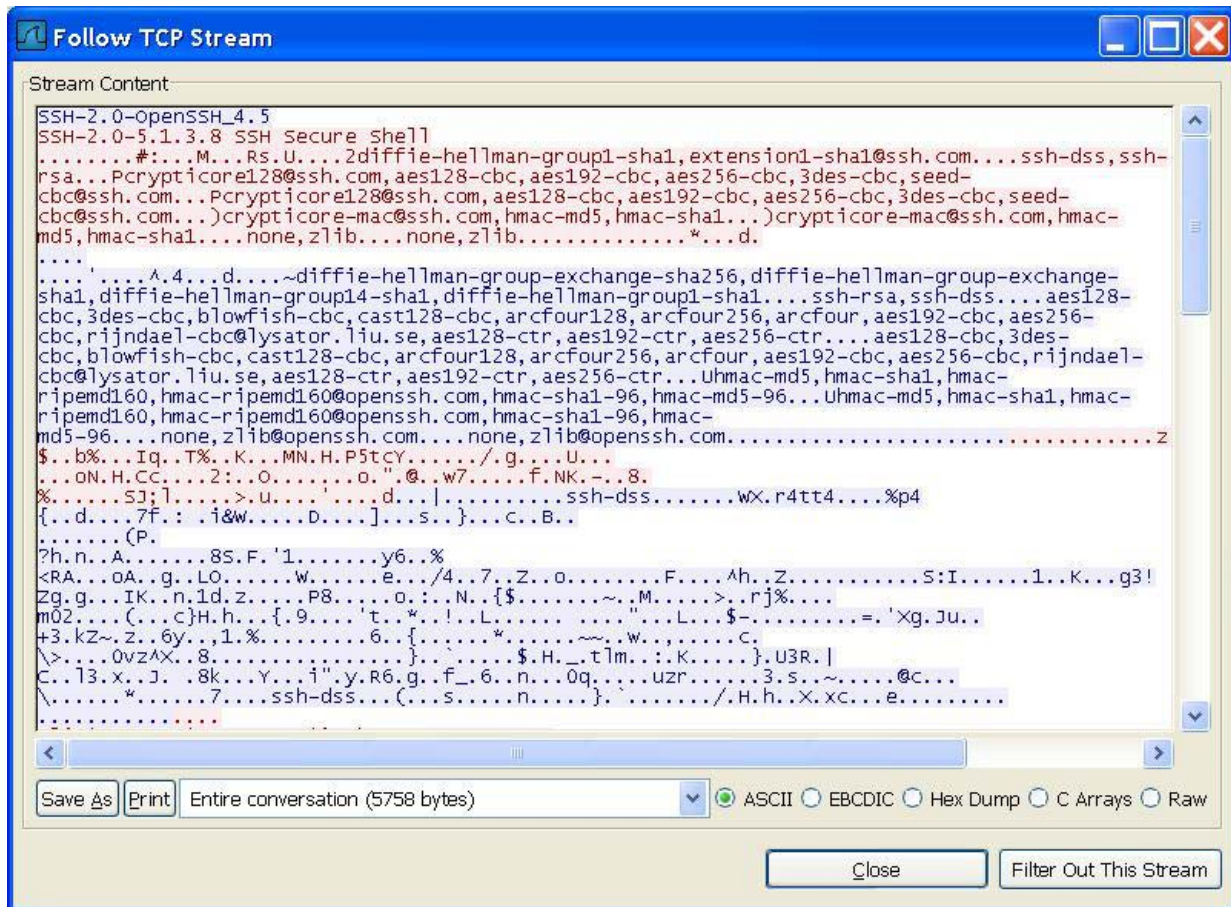
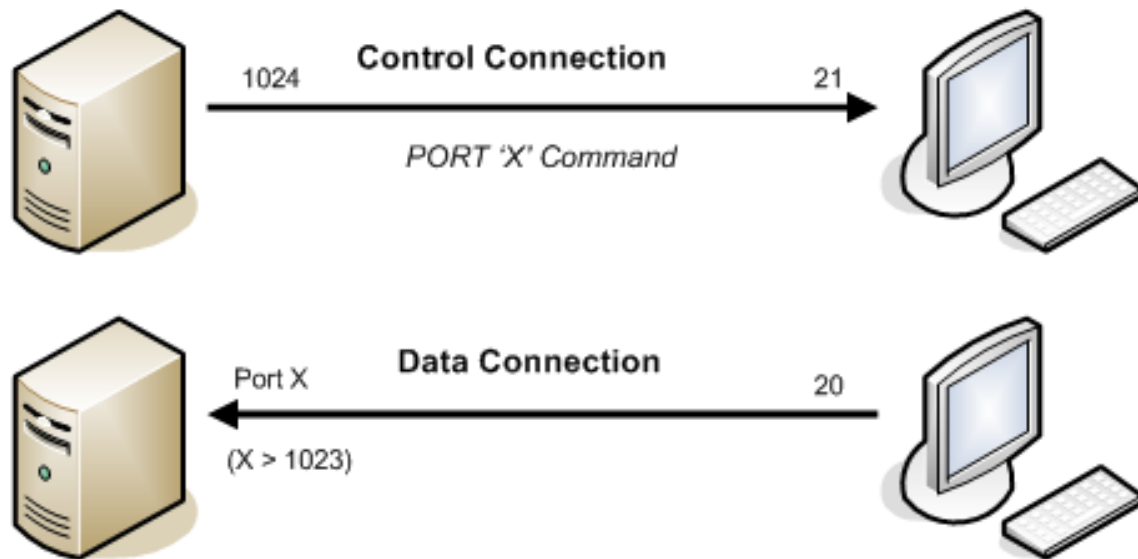


Figure 1.2: An example of Secure FTP, which is much less informative about its transaction.

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.



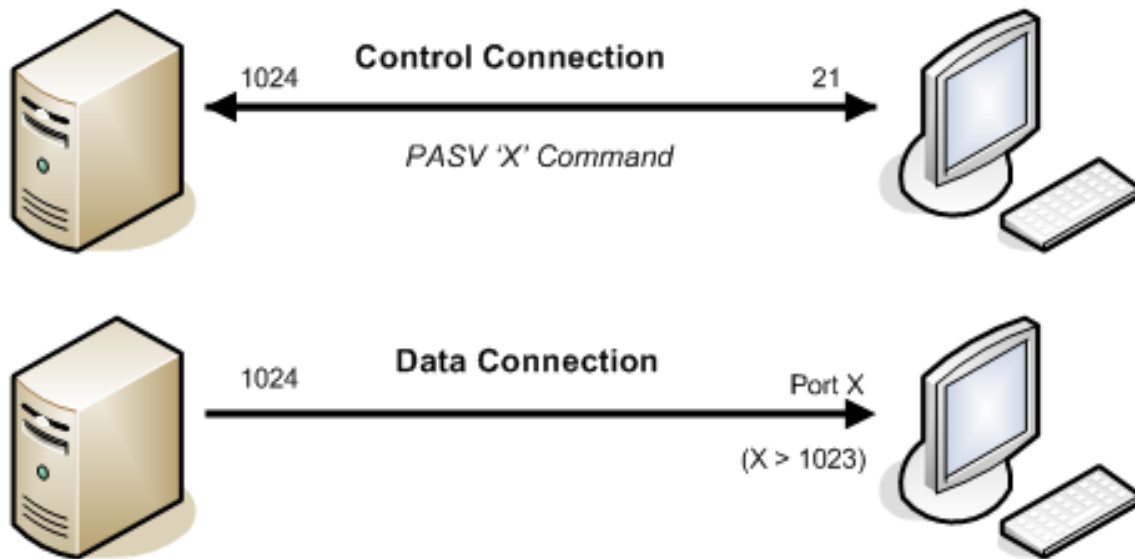
## Active FTP Connection



*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 16<sup>th</sup> 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.

## 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>.