Realtime publishers

The Shortcut Guide™ To

Secure, Managed File Transfer

sponsored by

IPSWITCH FILE TRANSFER

Don Jones

Copyright Statement

© 2010 Realtime Publishers, Inc. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtime Publishers, 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 Realtime Publishers, Inc or its web site sponsors. In no event shall Realtime Publishers, 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.

Realtime Publishers and the Realtime Publishers 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 Realtime Publishers, please contact us via e-mail at info@realtimepublishers.com.

[**Editor's Note:** This eBook was downloaded from Realtime Nexus—The Digital Library for IT Professionals. All leading technology eBooks and guides from Realtime Publishers can be found at http://nexus.realtimepublishers.com.]

Chapter 2: Common File Transfer Myths

As I work with consulting clients and as I speak with IT professionals at various conferences and tradeshows, I encounter more than a few misconceptions and bad assumptions related to file transfer. Some of these myths range from relatively minor misunderstandings to extremely major beliefs that actually hold back the person's entire organization. Let's play "Mythbusters" and examine some of these myths. I'll look at the most common ones I run across, explain where they came from—because many of them do, in fact, contain a nugget of truth—and see how they hold up to cold, hard facts.

Myth 1: Security Is Not Important

This is probably one of the first and most wildly inaccurate myths I run across. I can barely comprehend anyone in a modern business environment believing that security really isn't important. Today's businesses *know* that security is important—and therein lies the grain of truth in this myth. *Today's* businesses *do* care about security; businesses of yesteryear often did not. In fact, even through the late 1990s, many businesses simply didn't focus very much on security. It wasn't at all unusual for file servers to contain a permission for "Everyone: Full Control" at the root of their hard disks, with that permission inheriting to every file and folder on the server. I worked for one company in the late 1990s that assigned a public IP address to every computer on the network and didn't have a firewall between their network and the Internet. Unthinkable a decade later, but in the past, there simply weren't quite as many security threats, and so there wasn't much security focus. In many respects, security *wasn't* important, at least not for many businesses.

If Security Isn't Important, Then Why…

I enjoy having conversations with clients who start by saying, "security isn't really a concern for us." In most cases, what they're telling me is that they don't want to use security as a blanket argument in favor of implementing some technology solution—they want other reasons to implement something. Which is fine. But prior to having that conversation, I usually had to check in with a security receptionist, sign into a log, be issued a guest badge, get escorted through card key-protected doors into a conference room, and had my route monitored by security cameras. And "security isn't a concern" for them?

I think "security" gets pulled out as a business driver so often that business people—especially business technology people—just get sick of it. It seems like every IT vendor in the world tries to use "security" as a way of getting their foot in the door or closing the sales pitch. And that can definitely be frustrating, but it's disingenuous to say that "security isn't a concern for us."

Trite as it is, security *has* to be a concern. It's rarely the *only* concern, but it's always going to be there. I've never met a single company who could happily live without any security concerns at all—everyone locks the doors to the office, keeps the cash in a safe, and so on. IT security is no different—we all get tired of hearing about it and reading about it, but it's something we have to pay attention to.

Today, of course, the world is different. There are a few more security threats out there, and we've become *aware* of many more security concerns than we were in the past. Nobody in their right mind would operate a network without one or more firewalls, without anti-malware tools in place, and so on. There are really two reasons that today's companies focus a bit more on security than they did in the past: internal concerns and external requirements.

External requirements are fairly new, coming into play in the mid- to late-1990s. These are often imposed by governments or by industry groups, typically focused on a single business industry or class of companies and are often designed to bolster consumer protections or government oversight. In the US, common external requirements include:

- The Payment Card Industry (PCI) Data Security Standard (DSS)
- The Health Insurance Portability and Accountability Act (HIPAA)
- The Gramm-Leach-Bliley Act (GLBA)
- The Sarbanes-Oxley Act (SOX)
- Various US federal requirements for government agencies and contractors

At a very high level, these requirements have some common goals. Each of them focuses on a particular kind of data; HIPAA covers patient healthcare information, for example, while SOX focuses on the financial reporting of publicly-traded companies. For the information that these requirements focus on, they typically require that:

- Information not be disclosed to unauthorized entities
- Access to information be audited in a permanent log
- Information be verifiably secured to prevent unauthorized disclosure
- Information be retained for some specified period of time (and often no longer)

Security Is Not Just Security

Security *per se* isn't what many companies are *most* concerned about, especially companies subject to external requirements. Former Gartner research director L. Frank Kenney said, "It's never just been an issue of security, although security is one of those low-hanging fruits that get everyone's attention. More companies are affected by a failing audit and the fear of failing audits. The bigger issue is: Can I be assured, can I show, that a file that moves from point A to B has been secured, that the person who sent it had the authority to send it and was authenticated?" and so forth. *Security* is simply the broad term we use as a shorthand way of referring to all of these concerns.

Although the specifics, of course, differ between them, these common high-level goals make it clear that security *is* important—to someone watching over your company, if not to your company itself. These external requirements often come with fines for non-compliance, and can in some cases restrict or remove a company's ability to operate in a given industry. The PCI DSS, for example, is enforced by credit card companies, and failure to comply with its requirements can result in a company losing the ability to process credit card transactions—a pretty serious obstacle for many companies.

Internal requirements encompass all the things that a company does for its own benefit. In some cases, companies' internal requirements mirror and supplement those of external requirements—because, in many cases, the external requirements are just a codification of "doing what's right" in the first place. In other cases, internal security requirements help protect the company from things it perceives as damaging, such as industrial espionage and intellectual property theft and disclosure of customer (or other) information that would result in damaged customer relations.

So where does all of this fit in with file transfer? *Security* in the IT world nearly always refers to the security of *information*—the "I" in "IT." Information tends to leave a company in one of two major ways: via email and via file transfer. Simply put, if you're going to find a violation of internal or external security requirements, you've got even odds that file transfer will be behind the violation. From a business perspective, you should really have a couple of major security goals:

- Ensure that only authorized users can transfer files and that only authorized recipients receive them. As this is often impractical, you at least want to keep track of who sends what to whom so that you can take corrective actions if necessary.
- Ensure that only the *intended* recipient can access transferred files. In other words, you don't want unintended people snooping on your information transfers.

There are some other more subtle security goals, but I'll save those for later in this chapter. At a basic level, making sure only authorized entities have access to your information is the primary goal of security—and it plays a *strong* role in file transfer.

> **Myth Busted**
>
> Security *is* important, and today's companies care. File transfer is a major opportunity for security problems, and security has to be a major consideration in any file transfer scenario. In fact, security is *so* important, that you *shouldn't even be talking* to vendors who can't lead off the conversation with a rock-solid security story.

Myth 2: Homegrown Is Cheaper

Most companies have relied—at least briefly—on homegrown file transfer "solutions" similar to the following example:

```
set locus local                          ; Avoid K95 LOCUS popup

ftp rawhide.redhat.com /noinit /anonymous

if fail end 1 Connection failed

if not \v(ftp_loggedin) end 1 Login failed


ftp cd /pub/redhat/linux/rawhide/i386/RedHat/RPMS/

if fail end 1 CD to RPMS directory failed


set ftp dates on                   ; Preserve file dates

set xfer display brief             ; FTP-like transfer display

ftp type binary                    ; Force binary mode
```

```
set incomplete discard              ; Discard incompletely received files
set ftp collision discard           ; Don't download files I already have

if >= \v(version) 800205 set take error on

; Get the files...

mget libstdc++-* glibc-devel* ncurses-*

mget /except:{*-devel*} 4* GConf2* Glide3* LPR* MAKE* O* PyXML-* SysV* V*

mget /except:{{*-devel*} {a[bm]*}{anaconda*}{aspell-[a-z]*}} a*
mget /except:{{*-devel*} {balsa*}{bash-doc*}} b*
mget /except:{{*-devel*} {chromium*}{compat-[dgl]*}{cvs*}{cWnn*}} c*
mget /except:{{*-devel*} {db4-[uj]*}{ddd-*}{d[bd]skk*}{desktop-*}{dia-*}-
                         {docbook-style*}{doxygen*}} d*
mget /except:{{*-devel*} {eel*}{emacs-[el]*}{emacsp*}{epic*}{evolu*}{exmh*}} e*
mget /except:{{*-devel*} {festival*}{fonts-*}{freeciv*}} f*
mget /except:{{*-devel*} {g[ailnt]*}{gcc-[gjo]*}{gd[bm]-*}{gedit*}-
                         {gphoto2*}{gsl*}} g*
mget /except:{{*-devel*} {gaim*}{galeon*}{gated*}{gtk-engines*}{gtkhtml-*}-
                         {gimp-print-c*}{gimp-[d,0-9]*}} ga* gi* gt*
mget /except:{{*-devel*} {*.i686.rpm}{glade*}{glibc-[dp]*}} gl*
mget /except:{{*-devel*} {gnomem*}{*-game*}{*-user*}{*-pilot*}{*-audio*}-
                         {gnucash*}{gnumeric*}} gn*
mget /except:{{*-devel*} {ht[td]*}{im-*}{inn-*}{imap*}{isdn*}{itcl*}} h* i* j*
mget /except:{{*-devel*} {k[emnopsvW]*}{kde*}{klettres*}{ktouch*}-
                         {kakasi-*}{kappa*}{krb5-serv*}} k*
mget /except:{{*-devel*} {kde-i18n*}{kde[2gmtv]*}{kdeartwork*}-
                         {kdebindings*}{kdepim*}{kdesdk*}} kde*
```

```
mget /except:{{*-devel*} {kernel-[bBdsu]*}} ke*.i386.rpm
mget /except:{{*-devel*} {kmail*}{knm*}{knode*}{koffice*}{kooka*}{kppp*}-
                         {kstar*}} km* kn* ko* lp* ks*
mget /except:{{*-devel*} {libgcj*}{libgnat*}{libtab*}{lic*}{la[mp]*}{lftp*}} l*
mget /except:{{*-devel*} {m[cguxy]*}{mailman*}{man-pages-[a-z]*}{mew*}-
                         {miniChin*}{mod_*}{mrtg*}} m*
mget /except:{{*-devel*} {nautilus*}{ncpfs*}{nmh-*}{noatun*}{nss_*}{nut-*}-
                         {nvi-*}} n*
mget /except:{{*-devel*} {*.i686.rpm}{octave*}{open[ho]*}-
                         {openldap-[cs]*}{openmotif2*}{openssl0*}} o*
mget /except:{{*-devel*} {p[hvwx]*}{pan-*}{perl-PDL*}-
                         {post*}{pydict*}{python-d*}} p*
mget /except:{{*-devel*} {qt-design*}{qt2*}{quanta*}{recode*}{ruby*}} q* r*
mget /except:{{*-devel*} {s[eqwy]*}{sane*}{skkd*}{snavig*}{splint*}{stard*}} s*
mget /except:{{*-devel*} {sendmail-doc*}{sylph*}{swig*}} se* sw* sy*
mget /except:{{*-devel*} {t[oW]*}{t*fonts*}{tclx*}{tetex*}{timidity*}-
                         {tripwire*}{tuxracer*}} t*
mget /except:{{*-devel*} {unixODBC*}{uucp*}{vim-[eX]*}{vnc*}} u* v*
mget /except:{{*-devel*} {w[3l]*}{wine*}{wordtrans*}} w*
mget /except:{{*-devel*} {x[aef]*}{xc[dhi]*}{xine-*}{xmms*}{xpdf-[a-z]*}-
                         {xsane*}{xtrace*}} x*
mget /except:{{*-devel*} {XFree86-[cdIX]*}} X*
mget /except:{{*-devel*} {zebra*}{zsh*}} y* z*


end 0
```

This script automates a command-line File Transfer Protocol (FTP) client. In this case, it's retrieving OS updates from a vendor's servers. It works. It isn't terribly pretty, but it works. Scripts like this have existed since the advent of FTP, and they will probably continue to exist for years and years. Systems administrators rely on them to automate complex tasks—clearly, the task automated by this script is fairly complex, and you certainly wouldn't want to type all those commands manually on a regular basis. And this is what FTP scripts commonly look like: A series of complex commands that execute in a sequence.

But is a homegrown "solution" like this one cheaper than a commercial file transfer application? It depends on what you mean by "cheaper." A homegrown solution is certainly cheaper than running the same commands manually; the time saved by the administrator is probably pretty easy to figure out. There's no question that automation saves time.

But a script like this can be tricky. FTP itself doesn't support a great deal of diagnostic logging; if a single command fails, you won't necessarily know about it. If your script is implementing a mission-critical business function—such as transferring data to a business partner on a regular basis—you may not know a problem exists until someone calls and asks where there data is.

Scripts like this can also be expensive to maintain. It's great to have someone on your team who has the knowledge and skills to write such a script. If they're the *only* one on the team, you'd better keep them happy, because if they leave, all of a sudden your convenient scripts become a huge liability. If one stops working, you may not be able to fix it quickly—meaning your business may suffer.

Homegrown = Very Expensive

Homegrown solutions aren't limited to FTP scripts. Sometimes companies accidentally put themselves into the software development business to solve a particular line-of-business problem in their own specific way. Hewlett-Packard did so, using an in-house system named Omega to keep track of commissions for the company's salespeople. Omega was the perfect example of a homegrown solution: It was completely outside the company's core expertise, was developed more than a decade ago (it started at DEC), and probably did a great job in the beginning and was never something the company intended to sell. But in 2009, the company was hit with a class-action lawsuit because the homegrown solution had improperly compensated some 2000 salespeople. The problem was simply that HP was bigger than Omega could handle, despite efforts to keep the software updated.

Christopher Cabrera of Xactly Systems says, "That's one of the big problems with homegrown systems: they can cost you big time, often just when you need them the most. In terms of hard dollars, homegrown systems are pricey to build, costly to maintain, difficult to economically scale, and expensive to modify in the face of business change. In terms of opportunity costs, they lack the latest features and functionality, and they slow down and mess up vital business processes and initiatives."

Getting back to file transfer, an *Information Security Magazine* article quoted Gartner analyst L. Frank Kenney, "Most places have homegrown solutions. It's not greenfield; just about everyone has leveraged FTP." The article also noted that most industry experts believe, "If you roll your own, the cost is very high. And, you don't have a consistent way to manage security around file transfers."

Often, homegrown solutions start with a pretty simple objective: move this file from here to there. Then one day a problem occurs, and it takes a few days for anyone to notice. So the homegrown solution is modified to include some logging and perhaps email notifications of success or failure. Then some security auditing is added. Then the script is modified to support encrypted connections. Server names change, so the script is modified to have the new connection information. An OS upgrade breaks the script, so it's rewritten. Over time, without realizing it, you've spent a lot of time and money on this homegrown, "cheaper" solution. Your company is now in the business of application development and support—albeit in a part-time, unintended fashion.

My point is this: Homegrown solutions *seem* cheaper at the start. They rarely stay that way, and their costs creep up on you. Homegrown tools rarely meet *every* business need—did you see any security auditing in that FTP script? Any reporting? Anything to prevent use by unintended individuals? I have *never* been to a consulting client that didn't have some kind of homegrown file transfer tools in place, and I have *never* spoken with a client who didn't regret, to some degree, having such a strong dependency on those homegrown tools.

Myth Busted

If something is important to the business, then it should be done using business-grade tools, not homegrown hacks. Unless your company writes file transfer software for a living, you shouldn't be writing file transfer software—like scripts—at all.

When Homegrown Isn't Bad

If you've ever looked at a commercial file transfer solution, you know many of them support some level of automation and customization—often through scripts. So why aren't *those* scripts the same as the "bad" ones I've been discussing?

A "plain" FTP script starts with very little functionality beyond moving files back and forth—it's not a full programming language, there's no logging, no security, none of that. Adding those capabilities takes even more programming, and that's where you start really spending money on something that you thought was "free."

A file transfer solution, however, *has* all those capabilities built in (assuming it's a decent one, of course). A "script" that runs *within* that solution is simply invoking pre-built capabilities and functionality in a specific order—literally a script, telling the actor (the file transfer software) what lines to read (what actions to take) in a specific order. Those scripts aren't as hard to maintain over time because they tend to be less complex.

A *really* good file transfer solution, however, will allow you to "write scripts" without writing scripts. Some offer graphical user interfaces and "wizards" to help build automated task sequences. Behind the scenes, the result might indeed be a script, but you'll be maintaining it in a way that requires fewer specialized skills and less effort, so you'll keep your overhead lower.

Myth 3: File Transfer Is Just FTP

One reason that many companies start with homegrown tools for file transfer is the perception that "file transfer" is nothing more than FTP, and that in many cases it's a scheduled, automated FTP between two servers—such as transferring data on a recurring basis to or from a business partner. In the 21st century, though, FTP isn't the only game in town. First, there are the numerous variants of FTP that I mentioned in the previous chapter: FTPS, Secure FTP, SFTP, and so on. There are also network copy protocols, such as Server Message Blocks (SMB), Common Internet File System (CIFS), and so on. Even email has become a means of file transfer by simply attaching files to email messages.

You don't always get to pick which file transfer technique or protocol you use. You might be happy using FTP because it's easy to automate with those homegrown scripts; a new business partner, however, might insist that data be transferred using the HTTPS protocol—meaning all your FTP scripts are useless. Another new business partner might only want to send data via email attachments—again negating all your FTP skills. Another business partner might require you to transfer data according to the AS2 specification—and your on-staff FTP jockeys might not even know what that is (it's a combination of HTTP and S/MIME) let alone how to crank out a script for it.

In today's business world, flexibility pays. If your business wants to transfer data to a partner, and that partner requires the use of something like AS2, which answer would *you* rather give your executive team?

- "Um, no, we don't even know what that is, let alone have the ability to write a script to do it."
- "Sure thing, boss."

One reason that file transfer has moved beyond FTP is that FTP is a fairly primitive technology. It was invented in 1971, after all, and despite numerous updates over the years, it doesn't provide a lot of the features folks needs these days—like built-in encryption, delivery confirmation, security logging, content encoding, and so forth. FTP still has its place, as it's fairly simple to use and is available on almost every computer OS in existence—but it isn't the only game in town.

> **Myth Busted**
>
> File transfer can be a *lot* more than FTP, and you won't always be able to force FTP as the file transfer solution for a given scenario.

Myth 4: Email Is Safe and Secure for File Transfer

This is a myth I've spent a long time arguing about. Let me first acknowledge that of course an email can be encrypted, digitally signed, and so forth, and that most email systems support delivery confirmations and other feedback mechanisms. As a means of moving data from one place to another, email is simple to use, fairly reliable, and broadly accessible. That *does not mean* it is "safe and secure" for business-critical data. True, sometimes email might be your only option (if a business partner insists on it, for example), but that doesn't mean email isn't without significant downsides.

File Transfer Isn't Just for Servers

In a book like this, it's very easy for me to fall into the pattern of talking about system-to-server transfers. That's the type of file transfer I typically work with most, as many of my clients have hired me to automate regularly-occurring transfers, typically between one of their servers and a server owned by one of their business partners.

System-to-system file transfers are, for me, easy. You plop a file transfer solution in place, set up connection parameters, pick a file transfer protocol, and set a transfer schedule. Everything happens in the background, and nobody but an administrator has any access to change things or mess things up.

In an *Information Security Magazine* article, John Thielens was quoted as saying, "Ad hoc file transfer is an important trend. You think of managed file transfer as something scripted or for techies, but there's also file transfer technology for human-to-human collaboration."

It's true. Those system-to-system transfers are a tiny fraction of a company's total file transfer volume. *Most* data is moved around by end users on an ad-hoc basis, most often in the form of email attachments. I can't very well expect end users to log into the corporate file transfer solution, set up connection parameters, specify a transfer schedule, and so on—those end users don't have the knowledge to do so, would probably mess up other people's transfer jobs, and quite frankly don't want to go through all that hassle just to get a spreadsheet over to a business partner. That's why email attachments are so popular.

So why is email not "safe and secure?" Because of the way email functions. Look at this:

```
To:              Don Jones <securitynews@xyz.com>

Delivered-To:              securitynews@xyz.com

Received:              by 10.150.220.11 with SMTP id s11cs108046ybg; Mon, 28 Dec
2009 12:56:39 -0800 (PST)

Received:              by 10.142.2.14 with SMTP id
14mr10754165wfb.15.1262033799234; Mon, 28 Dec 2009 12:56:39 -0800 (PST)

Received:              from mail.wingateservices.com (mail.wingateservices.com
[68.142.139.11]) by mx.google.com with ESMTP id
6si89481422pzk.103.2009.12.28.12.56.38; Mon, 28 Dec 2009 12:56:38 -0800 (PST)

Received:              from rsa.wingateservices.com (rsa.wingateservices.com
[172.17.8.10]) by mail.wingateservices.com (8.13.1/8.13.1) with ESMTP id
nBSKubrk032469 for <securitynews@xyz.com>; Mon, 28 Dec 2009 13:56:37 -0700
```

Those are the full message headers—something your mail client normally hides—from an email message I recently received (although I've changed the email addresses and server addresses for this example). This was hardly a "point to point" transfer; the headers show that my message traveled through something like four different mail servers before it got to me. Here's another:

```
Received:              from smtpin137-bge351000 ([unknown] [10.150.68.137]) by
ms214.erp.com (Sun Java(tm) System Messaging Server 7u3-12.01 64bit (built Oct 15
2009)) with ESMTP id <0KVF00BPX2UW4630@ms214.mac.com> for john.doe@abc.com; Tue,
29 Dec 2009 05:59:20 -0800 (PST)

             Received:              from mail.abc.net ([unknown]
[64.142.73.206]) by smtpin137.erp.com (Sun Java(tm) System Messaging Server 7u2-
7.04 32bit (built Jul  2 2009)) with ESMTP id <0KVF0065Y2UVZN90@smtpin137.mac.com>
for abc123@erp.com (ORCPT abc123@erp.com); Tue, 29 Dec 2009 05:59:20 -0800 (PST)

             Received:              from server75.appriver.com
([207.97.224.142]) by mail.abc.net with Microsoft SMTPSVC(6.0.3790.3959); Tue, 29
Dec 2009 05:59:17 -0800

             Received:              from [10.238.8.51] (HELO
inbound.appriver.com) by server75.appriver.com (CommuniGate Pro SMTP 5.2.13) with
ESMTP id 954687680 for john.doe@abc.com; Tue, 29 Dec 2009 08:59:10 -0500

             Received:              from tippit.wc09.net ([74.203.49.55]
verified) by inbound.appriver.com (CommuniGate Pro SMTP 5.1.7) with ESMTP id
641729304 for john.doe@abc.com; Tue, 29 Dec 2009 08:41:39 -0500

             Received:              from aweb06.whatcounts.com (172.16.2.16) by
tippit.wc09.net (PowerMTA(TM) v3.5r15) id h783j60kup8j for <john.doe@abc.com>;
Tue, 29 Dec 2009 05:14:12 -0800
```

This time, the message went through six servers before it got to its destination. That's how email works—in fact, it's a major feature of the Simple Mail Transfer Protocol (SMTP) that powers Internet email. The idea is that mail can take many routes to get to its destination, so if one route is down, the mail can still get through.

That's all well and good when your email is a message to Mom, but when you're talking about sensitive business information, email's roundabout paths present some distinct disadvantages:

- All this bouncing from server to server takes time. How often have you been on a conference call, sent an email to someone else on the call, and then waited—patiently, I'm sure—while they hit "refresh" over and over on their mail client? Critical business information may need faster, point-to-point delivery.
- Email doesn't actually support true delivery confirmation. In other words, you the sender can't hand the email to the recipient and know that it's in the recipient's hands. What email can do—if the recipient's email system supports it—is send another email back to you letting you know they got it. Those confirmations can obviously be forged, mislaid, misrouted, or eliminated en-route. Direct, point-to-point transfers provide actual delivery confirmation.
- Your email messages pass through many hands, and each server on the route can keep a permanent copy. In practice, most legitimate mail servers never do because of the space that would entail. But they can. And even if you've encrypted the email contents, giving someone a copy of the message will give them time to break that encryption. Older 40-bit encryption—which is still in widespread use—can be broken by a home computer in a few days. You'll never know that the decryption is in progress because a copy of your email will have been forwarded to its intended recipient.
- Even encrypted email messages contain certain information that might be considered sensitive. For example, the sender and recipient identities can't be encrypted because that information is needed to deliver the message. Anyone intercepting the message—whether it's encrypted or not—will know that those parties are in discussions. That knowledge helps someone involved in industrial espionage, for example, focus their efforts on the most promising messages.

For companies that spend thousands of dollars on card keys for their office doors; tens of thousands on computer security like password management tools, firewalls, and so on; and thousands more on security cameras—well, it's just amazing that those same companies would ever consider transferring sensitive data via email. You might as well print the sensitive information, make a few photocopies, stick the copies in envelopes marked "please do not read," and put them in the magazine rack in the office's lobby.

Email sees heavy use as a file transfer mechanism because it's usually the only thing your end users have that is broadly accessible and is compatible with just about every other company in the world. And, as I acknowledged earlier, sometimes it's the only way two parties can exchange files with one another. That does not, however, mean it's safe and secure, and it doesn't mean you shouldn't be looking at some other, better solution for ad-hoc, person-to-person transfers.

Myth Busted

Email is *not* as safe and secure as a point-to-point transfer that doesn't involve so many intermediaries. Email is convenient—but you can look for safer solutions that offer similar convenience.

Myth 5: Slick Means Functional

This is something I'll look at in more detail in Chapter 4, but for now, let's just acknowledge that the best-looking solution isn't always the most-functional solution. A pretty graphical user interface is nice, but when it comes to file transfer, it's what's under the hood—protocol support, security features, logging and auditing, stability and reliability, and so on—that matters most.

Myth Busted

Great-looking file transfer solutions *can* offer great functionality—but more utilitarian-looking solutions can also offer great functionality. In the end, you should choose a solution based on features and capabilities, not on visual appeal.

Myth 6: All Encryption Is Equal

There was a time—back in the mid-1990s, probably—when 40-bit encryption was the king of the security world. Today, as I've mentioned, a typical home computer can crack 40-bit encryption in a few days using brute-force methods. Nobody with any serious concern for privacy would rely on 40-bit encryption today—but you'll still see software encryption packages touting 40-bit encryption support.

Note

The only exception, and the right time to use 40-bit encryption, is when it's the only option for legal reasons. Some countries prohibit the use of strong encryption without a special permit, and some extremely strong encryption algorithms can't legally be exported to certain countries, meaning companies in those countries don't have access to stronger encryption.

Slightly stronger 56-bit encryption is little better; in 1998—more than a decade ago—a specially-built hobbyist computer was cracking 56-bit keys in a few days, and could decrypt 40-bit encryption in a few seconds.

As a general rule, today's common encryption algorithms rely on 128-bit, 192-bit, and 256-bit encryption keys; larger keys are possible but are rare. These key sizes all refer to symmetric encryption, which is one broad system of encrypting data. But key size isn't the only factor in encryption—the algorithm, or the way the key is used, also plays a role. For example, a 1024-bit key used in an asymmetric system such as RSA is considered equivalent, from a security perspective, to an 80-bit key in a symmetric system; 15,360-bit RSA keys are considered equivalent to a 256-bit symmetric key.

Resource

A complete discussion of the details is beyond the scope of this book, but if you're an encryption buff, you'll find http://en.wikipedia.org/wiki/Key_size to be a good starting point for further reading.

The point of all this is that all encryption *isn't* equal. That's not to say your company needs the security of a 512-bit symmetric key, which is what the government requires for "Top Secret" information. However, you may well need just that. So as you look at file transfer solutions, make sure you're examining—in detail—their encryption capabilities. Simply offering encryption isn't enough; you need to know what *kind* of encryption.

One benchmark you can look for is Federal Information Processing Standards (FIPS) 140-2 approval and validation. FIPS is maintained by the US National Institute of Standards (NIST) computer security division. FIPS are used by both the US and Canadian governments, and serve as guidelines to other governments as well as many private-sector security specialists. FIPS 140 covers cryptographic modules; FIPS 140-2 is (as of this writing) the most recent and stringent version of the standard (FIPS 140-1 was written in 1994 and has not been in use since 2003).

Validation is the word to look for. Software developers can claim that their products are "written to FIPS 140-2 standards," but unless they've been independently tested to verify the claim, it's just words on paper. NIST co-administers a validation program, called the FIPS Cryptographic Module Validation Program (CMVP—you knew there was another acronym coming), which is used to test software for compliance with the FIPS 140-2 standard. The program isn't just an automated test of the software, either; it includes a review of the product's design document and source code, as well. Validated software has been confirmed to use the approved cryptographic algorithms, reviewed to ensure there are no "back doors," and examined to ensure that sensitive data is securely erased when it is no longer needed. Software that passes the complete review is issued a numbered certificate, which a software developer should be able to produce for you.

Other FIPS you may run across include FIPS 197, which covers the Advanced Encryption Standard (AES) encryption algorithm, and FIPS 180, which covers the SHA-1 and HMAC-SHA-1 encryption algorithms. Simply put, these standards—which cover government-grade security—can serve as an easy way to determine whether a product offers solid, validated encryption.

Where Was Your Encryption Made?

Use some caution when evaluating file transfer software that offers FIPS-level encryption. Achieving FIPS certification is an expensive and time-consuming process; some software developers prefer not to invest in it and instead purchase a pre-made, FIPS-compliant cryptography component from a third party. That's fine, but you need to ask some detailed questions and examine the software design to be sure that the integration between the file transfer software and the third-party cryptography module is solid and secure. Whenever possible, I prefer software that's a little more end-to-end, at least when it comes to security, and to work with a vendor who has tightly integrated their cryptography code with their main software code. Especially when it comes to security, the "gaps" that can exist when integrating a third-party software module make me nervous.

Resource

You can read more about NIST, FIPS, CMVP, and more than nine other acronyms at http://csrc.nist.gov/groups/STM/index.html.

Myth Busted

Clearly, all encryption is *not* equal or FIPS wouldn't exist. If data privacy is important to you, make sure you understand exactly what kind of encryption you're getting with a particular solution.

Myth 7: Security Is Just Encryption

This is usually the argument I get in response to my assertion that email isn't a safe, secure means of transferring files. "But if we encrypt it, then it's safe and secure." That's kind of true, as far as it goes. If you use, say, 256-bit encryption on an email message, and you aren't concerned about timely delivery, then it doesn't matter if someone intercepts your email—they're probably going to require a few years, at least, to break the message. Fun studies have been done to guess how long it would take to crack encryption that used various lengths of keys—http://en.wikipedia.org/wiki/Brute_force_attack has some of the more impressive numbers, like the estimate that a 128-bit key would take something like 10^{13} years to crack using brute-force techniques. For the sake of argument, let's just say that 256-bit encryption is, using current and foreseeable computing technology, invulnerable—something like $3x10^{51}$ years would be needed, and likely whatever was in the message would be pointless after all that time. So does that negate my argument about email not being a safe and secure file transfer mechanism?

No. Encryption, and the privacy it delivers, is not the sum total of security. Security is *much* more. In fact, as I noted earlier, privacy is not always the top security concern most businesses have.

Encryption protects you from eavesdropping, and that's important. But it will not stop someone from willfully releasing information to the wrong entity. Encryption won't keep a log of who sent the file. Encryption won't prove that the file actually got to where it was going. These are also security concerns, and if you're dealing with business-class file transfer, they're capabilities you *need.* Let's take them in turn.

Non-Repudiation and Guaranteed Delivery

"I sent it."

"Well, I never got it."

"Yes, you did."

"Well, it was different than what you sent."

That's a conversation involving *repudiation*, or denying something—in this case, that a file was successfully delivered. There are certain types of business information that *must* be delivered, and you need to have proof that the delivery was completed—proof that can stand up to a possible denial by the recipient. In the physical world, we rely on express carrier tracking numbers, postal delivery receipts, and other devices; even then, it's hard to prove that a document was tampered with in transit, and almost impossible to prove the sender's identity.

So there's more to non-repudiation than just proving delivery: Ensuring that the document that was sent is in fact the same document that was received—that it wasn't tampered with in transit. An aspect is the recipient's ability to validate the sender's identity—critical for ensuring that received documents are authentic and not forgeries.

Typically, all of these needs are realized through the use of digital signatures, which are created from asymmetric-key encryption. In an asymmetric key, two actual encryption keys exist, and they are different. If something is encrypted with one of the two keys, only the other one can decrypt it—and vice-versa. One key is typically kept private by the key holder—and is appropriately called the *private key.* The other, *public key* is readily available to anyone who needs it.

Here's how it works: The sender of a file uses a private key to digitally sign a file. This digital signature is a small piece of information that is encrypted using the private key; typically, the signature includes identity information about the sender, and checksum information that can be used to verify the file's integrity. The recipient obtains the sender's public key and attempts to decrypt the signature. If the decryption is successful, the sender's identity is verified—because only the sender, using the private key, could have encrypted the signature in the first place. The decrypted signature's information—such as a checksum—can be used to verify that the file itself was not changed in transit. This kind of cooperation between sender and recipient requires specialized software on both ends of the file transfer.

As always, there are fine details that you need to be aware of. Although cryptographic signatures are fairly standard, you should ensure that they use a high level of encryption to prevent tampering—anything FIPS 140-2 compliant will do the job. Also, the checksum used to check the integrity of the transferred file is crucial. Older integrity checks use a cyclic redundancy check (CRC—you'll also see CRC-32 or XCRC, two more recent variants). Those aren't ideal because they're fairly easily fooled. A better approach is to use a cryptographic hash, such as a SHA1 or MD5 hash. These produce a more reliable and tamperproof "fingerprint" for a file.

"Guaranteed delivery" also involves encryption. The recipient generates some kind of "delivery confirmation" and applies a digital signature to it. This allows the sender to further validate the recipient's identity, and gives the sender undeniable evidence of the successful delivery.

Auditing

Who is sending files is a big deal, especially if your organization is dealing with external security requirements and is subject to audits of your security practices. *Auditing* is a word that's actually used in a couple of ways:

- It's the act of reviewing security logs and practices, typically by a human being who is checking for compliance with various regulations or requirements.
- It's the act of saving security information to a tamper-evident or tamper-proof log, for use by those human auditors; this activity is more accurately referred to as *logging.*

File transfer solutions obviously don't do the first kind of audit, but they certain can and should do the second: Logging file transfer activity to a tamper-proof or tamper-evident log. Commonly, that log will either be some kind of proprietary database or an external relational database management system (such as Oracle or Microsoft SQL Server), and the anti-tamper techniques usually involve more encryption.

There's a certain amount of trust that you have to place in the software. Obviously, unless you *wrote* the software, you can't really *prove* that it is logging *every single action*; you have to trust that it is doing so, and perhaps perform some tests to demonstrate to yourself that everything seems to be logged. There should not be an easy way to disable logging once enabled, and the ability to disable logging should be separated from anyone who uses the file transfer solution and might have cause to hide their activity at some point.

Authorization

Authorization is all about who is allowed to transfer files, what kinds of files they're allowed to transfer, when they're allowed to transfer, and what transfer options—such as whether they want encryption—they can specify.

This is obviously crucial for any kind of security concern. A good file transfer solution will understand this, and give you the ability to specify that *these* groups of users can perform *these* kinds of file transfers, and so forth, customizing the solution to fit our needs.

Retention

Retention is becoming increasingly important when it comes to file transfer. Typically, a file transfer solution needs to keep a copy of a file on-hand until the transfer is complete. Whether it is receiving or sending a file, the file is going to exist—possibly in an incomplete state, if it's being received—on the file transfer solution's hard drive, or in its memory, for some period of time.

You should be able to specify rules that govern this retention. For example, you may want the solution to move completed, received files to an alternate location, then securely erase the solution's own copy of the file. You might want sent files to be securely erased from the solution as soon as the transfer is successful. However, you might want received files to be retained—as a form of backup—for several days, even if the files were moved elsewhere by an automated process.

The important thing is to make sure that whatever you're using for file transfers can do what you want—ideally without you having to write a bunch of homegrown scripts to accomplish something your solution can't.

Coming Up Next…

Any technology—including file transfer—is only useful insofar as it helps solve some business problem or meet some business requirement. In the next chapter, I'll begin looking at business requirements, and mapping those to the technical capabilities offered by commonly-available file transfer solutions. We'll really focus in on what the *business* needs, and use those needs to build a sort of shopping list for file transfer capabilities. I'll get pretty detailed, because as you start evaluating file transfer solutions, I think you'll be able to do so more effectively if you can really compare detailed features and technical implementations, rather than just the high-level bullet points often listed in product brochures and Web sites.

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.