Global Information Assurance Certification Paper

The challenge of securely storing and transporting large files across a corporate Wide Area Network.

GIAC Gold Certification

Author: Jeremy Gibb, Jeremy@jeremygibb.com

Adviser: John Ives

Accepted: January 23rd 2007

Table of Contents

1. Introduction

The majority of organizations that use Wide Area Networks (WAN's) to connect Local Area Network's (LAN's) together have a requirement to transfer large amounts of data "across the wire", between different locations. A number of widely available desktop applications such as Microsoft Outlook and Windows Explorer provide built-in functionality that support the basic data transfer needs of most users (e.g. attaching a file to an email, creating a "share" on a remote machine and mapping a local drive to that share), but such solutions have limitations when there is a requirement from backend applications or system administrators to reliably transfer large files that are often numerous Gigabytes (Gig), or more, in size. This challenge is further complicated when the data is of a sensitive nature and needs to be transported securely, on a repetitive (i.e. automated) basis, and must be held in a secure format before and after transmission.

Government legislation such as SOX (the Sarbanes-Oxley Act, signed into US law July, 2002[1]), and HIIPA (The Health Insurance Portability and Accountability Act, enacted by U.S. Congress in 1996[2]), have forced companies to review a wide range of business practices, many of which include improving the security processes and procedures in place when handling and working with client data and other sensitive information. This, in part, has led to the requirement to encrypt data not only during transmission over the WAN, but also when it is stored on both client and server machines before and after transmission.

To further protect information within the organization, security professionals strive to implement solutions that meet all three of the fundamental security characteristics that form the CIA Triad, "a widely-used information assurance (IA) model which identifies confidentiality, integrity and availability as the fundamental security characteristics of information".[i]

---

[1] See http://www.soxlaw.com/
[2] See http://aspe.hhs.gov/admnsimp/pl104191.htm

Sensitive data may be compromised during transit over the WAN as a result of design vulnerabilities and weaknesses in the protocols being used to transfer the data, or through security vulnerabilities affecting the physical medium of the WAN itself. As a result, there is a requirement to identify and implement a secure data transfer solution that will meet the businesses data transfer needs.

2. The Un-trusted Corporate WAN

The continued growth and evolving need for mobile access to corporate networks and data, coupled with the implementation of cheap and flexible technologies like VPN's (Virtual Private Networks), has led to the "edge" of the corporate network becoming ever more distant and difficult to define. This change, coupled with existing weaknesses in WAN technologies, has increased the risk of the WAN being compromised, and the data transferred over it being exposed. Some of the factors that cause this increased risk include:

1. The shared medium over which most corporate WAN's are based; i.e. multiple customers using a common backbone infrastructure provided by the same Service Provider. Who has access to, and visibility of, corporate data when it is routed over the WAN? Is data is kept separate from that of other organizations using the same infrastructure. What guarantees are there that packet sniffing and data leakage (be it intentional or unintentional) does not occur?

2. The increased use of VPN technology over the past 10 years to provide cost effective and secure network connectivity over the Internet has created new concerns around WAN security and the data transmitted over them. Using VPN's to create secure virtual "tunnels" between multiple office locations over the Internet involves sending company data across numerous Internet Service Providers (ISP) backbones (albeit in encrypted format), most of whom the organization has no signed contract or non-disclosure agreement (NDA). Users with VPN client software on their laptops can connect back to the corporate WAN from un-trusted networks, often wireless, located in Internet cafes, coffee shops and hotel lobbies. As a result, organizations find their private data

being routed through third party network infrastructure over which they have no control or visibility.

3. The limited extent to which a network administrator or security officer can say they have complete understanding and accurate knowledge of their entire WAN, including all ingress and egress points to the Internet or other external client networks. Five or six years ago, when corporate Internet access that provided acceptable speeds (T1 and higher) required a lengthy ordering and provisioning process, such information was easier to keep track of (although even then it was not possible to prevent a user installing a modem in a desktop PC and connecting to the Internet over a normal telephone line using one of many, often free, local ISP's). Today, this challenge is even greater; proliferated by the onset of wireless broadband technologies offered by non-profit organizations to provide citywide free wireless Internet access to anyone wanting it. One such US city that is leading the way is Philadelphia, PA[3].

4. The WAN is only as secure as its weakest link. A company can commit unlimited amounts money and resource to designing, implementing and supporting a layered infrastructure that provides, to the best of our ability, a secure and trustworthy computing environment, but if a user in the Finance Dept decides he is unhappy with these controls and decides to connect to the Internet via one of the above described free citywide wireless Internet services, this will drastically reduce the level of security for the entire network, and dramatically increase the threat and risk to the data that is routed over it.

It is hoped that the technologies discussed in this paper will help mitigate the internal and external threats and risks associated with the transfer of data across the WAN, and the challenge of storing this data in a secure manner both before and after transmission.

[3] See http://philadelphia.wifi.com/

3. The Current Solution

Two of the most widely used data transfer solutions that come as standard with the majority of today's desktop applications are email attachments within Microsoft Outlook, and file shares within Windows Explorer. However, both have limitations when it comes to meeting the complex data transfer needs of backend applications and system administrators. These limitations primarily relate to the size of the files that can be transferred, and the inability of the technology to easily and reliably schedule future and repetitive transfers.

Email is unsuitable for the large scale movement of data across a WAN for a number of reasons. Email administrators often place limits on the size of files that can be attached to emails, and also the size of user's mailboxes. Most users find this inconvenient and consider it a hindrance but there is a reason for this. Email clients are not meant to be data stores or data transfer tools; they were simply not designed for this purpose. Stephen J. Bigelow notes "it's convenient for end users to attach a file, but those attachments can clog or crash an email server and grind business to a halt."[ii]

Other issues arise when using email to transfer data. By default, email is not encrypted. This means that unencrypted copies of attached files are stored in both the senders and receives mailbox, as well as being routed across the network in clear text format. This situation is exasperated when sending email over the Internet as system administrators have no visibility or control over the public network infrastructure (i.e. ISP owned and managed routers) that will be used to route the email message.

Windows file shares provide a simple and easy way for multiple users to share and exchange files across different machines, regardless of location. The main protocols that Windows file sharing uses are NetBIOS, Server Message Block (SMB) and Remote Procedure Call (RPC), however, none of these protocols are ideally suited to large scale, reliable and secure WAN data transfer. As noted by Timothy Evans in an online article that specifically discusses the Windows based file sharing protocols, "It is important to note the scale of LAN which NetBIOS was designed for. NetBIOS was not designed

for large networks".[iii] As a result, Windows file sharing is better used as a local solution, i.e. between two hosts on the same network.

One of the main reasons for this is because NetBIOS and SMB (which was modified and renamed by Microsoft to Common Internet File System (CIFS)) are "chatty" protocols, meaning that when a file is transferred between two hosts, a significant portion of the data sent is control data in addition to the data itself. Over a WAN where performance can be slow and packet loss reasonably high this chat contributes to poor performance. Such issues are noted in a document published on www.scps.org[4] "In IP VPN networks that go into and out of certain markets like China, where loss rates commonly exceed 5%, high loss rates can have a catastrophic effect on performance".[iv]

In the case of SMB, further performance issues are experienced because of its small default buffer size. This means that when transferring data via Windows file shares, each file must be split into numerous "blocks" before it can be transmitted, creating a requirement to send large quantities of packets between the client and server[5]. Because SMB uses a window based flow control system to control data transfer between hosts (i.e. so the sender will not transmit more data than the receiver can handle), for each block that is transmitted an acknowledgement must be sent from the receiver and received by the sender before the sender will send the next block. With this type of windowing system it is easy to see how even moderate WAN delays of 50-100ms will significantly slow the transfer of files that are only megabytes in size. This problem is made worse when packet loss on the WAN is factored into the equation as any dropped or lost acknowledgment packets mean the original block has to be resent. "These [CIFS] traditional file-sharing protocols were designed to share files over a local network, but perform poorly and prove unreliable in a WAN setting. Reading and writing files using standard file sharing protocols over WANs is orders of magnitude slower than over LANs."[v]

---

[4] SCPS is a protocol suite designed allow communication over challenging environments. Originally developed jointly by NASA and DoD's USSPACECOM to meet their various needs and requirements.
[5] See http://www.microsoft.com/windowsserver2008/branch-office.mspx

RPC operates (by default) over dynamic high level IP ports, making it difficult for the network administrator to control without configuring the firewall rulebase with unacceptable open rules. Microsoft notes that "problems with RPC can occur if your client is behind another firewall that blocks access to the dynamic, high-level ports that RPC connections require."[vi] Opening the firewall to these ports creates unnecessary risk within the network because malicious code contained within Trojans or self replicating viruses will often use UDP and TCP high ports as a transport mechanism or listening service[6] . This issue has been mitigated somewhat over recent years by software vendors redesigning their applications to run RPC over standard protocols such as HTTP which are generally permitted through most firewalls.

An alternate solution to email attachments and file shares is required. One of the most common and easy to use alternatives is File Transfer Protocol (FTP). FTP, as standardized by the IETF (Internet Engineering Task Force, set up to develop and promote Internet standards[7]) in RFC #0959[8] operates at the Transport network layer (Layer 4) of the 7 layer Open Systems Interconnection (OSI) model[9]. Being a TCP based protocol, FTP comes with built in support of ordered data transfer, retransmission of lost packets, error free data transfer and congestion and flow control.

FTP is widely available (one of a number of FTP servers available for free download is cuteftp[10]), supports both manual and automated file transfers and is a technology supported by most, if not all, desktop computer operating systems. In addition to numerous freeware, shareware and licensed products, FTP is supported by Microsoft's Internet Explorer and Mozilla's Firefox web browsers, and all UNIX and Windows Operating systems which have a built in client that is accessible

[6] One such example is documented at https://knowledge.mcafee.com/SupportSite/search.do?cmd=displayKC&docType=kc&externalId=VIL141166xml&sliceId=&dialogID=13418544&stateId=1%200%2013406956
[7] See http://www.ietf.org/
[8] See http://tools.ietf.org/html/rfc959
[9] See http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip
[10] See http://www.cuteftp.com/

from the Command Line Interface (CLI).

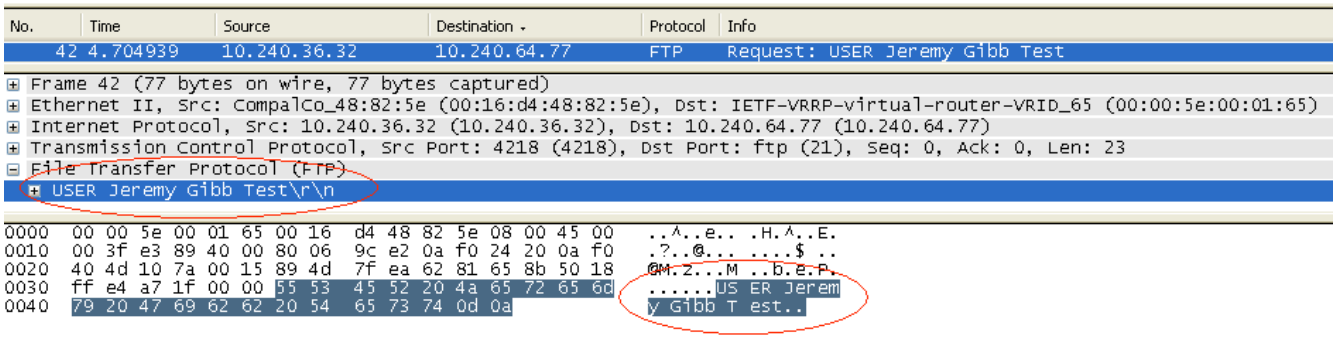
FTP is affected by a number of security vulnerabilities (enter "ftp security" into any Internet search engine and numerous results will be returned). The two main issues of concern to organizations wanting to securely transfer data over the corporate WAN are:

1. FTP transfers all data, including usernames and passwords used for authentication, in clear text. This means that anyone with physical access to the LAN or WAN could use network sniffer software (e.g. Ethereal[11]) or hardware (e.g. Fluke OptiView™ Series III Integrated Network Analyzer[12]), to intercept or eavesdrop the data transfer. As well as gaining free access to data, in instances where user credentials are also obtained, an attacker could use this information to gain further access into a users network profile (e.g. if AD credentials have been specified, the users email and potentially other network data is now at risk as users often use the same password across multiple applications and systems).

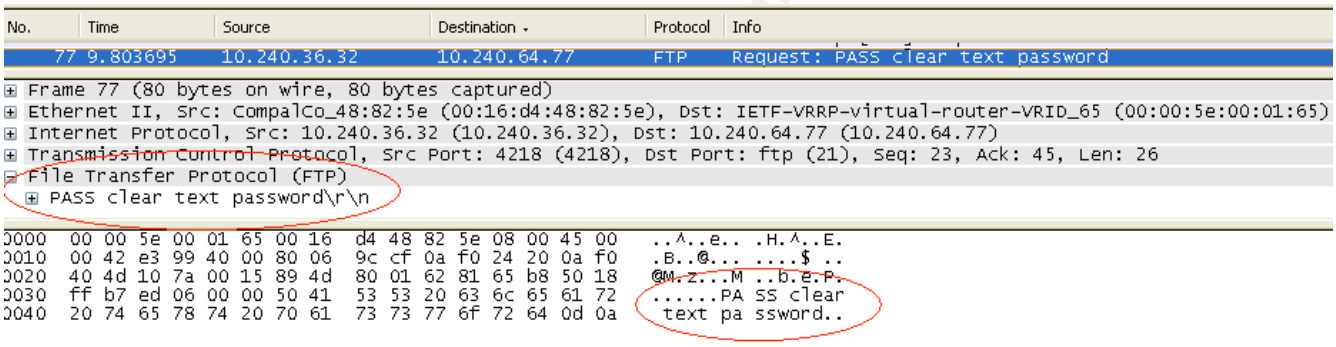
   Below is a screen shot from Ethereal, "the world's most popular network protocol analyzer"[13]. This open source freeware application allows anyone with physical access to any part of the network (i.e. between source and destination hosts) to "sniff", and subsequently analyze all data passed over it. To run this test, I downloaded the Ethereal software, installed it on my Windows XP laptop, ran it, selected "capture" and proceeded to connect to an FTP server on my network using the built in FTP client that comes with Windows XP.

   In the below, my laptop, the FTP client (source) has IP address 10.240.36.32. The FTP server (destination) is 10.240.64.44. Per the two circled fields, the username I used to log into this FTP server, "Jeremy Gibb Test", is clearly visible.

[11] See http://www.ethereal.com/
[12] See http://www.flukenetworks.com/fnet/en-us/products/OptiView+Series+III+Integrated+Network+Analyzer/
[13] See http://www.ethereal.com/



The second screen shot is taken from the same capture and shows the password I used to connect to the FTP server "clear text password".



The above information was obtained by sniffing the local Ethernet interface on my laptop. Imagine the information that could be obtained if it were possible to sniff all data going to (username and passwords) and from (the actual data itself) the FTP server.

Modern day switched networks are considerably harder than their hub based predecessors to sniff because unlike hubs, switches do not broadcast all data out of all ports. Physical access to the network no longer means instant access to all data. However, the ever growing availability of wireless Internet access points in locations like Starbucks coffee shops has once again increased the risk of data breaches through network sniffing as physical access to the network is no longer required. In fact, physical access to the building is also no longer required. Anyone with a wireless network card and sniffer software that supports that wireless card, who can get close enough to the

wireless network to pick up the transmitted packets (easy when the network is in Starbucks) can freely sniff all traffic. Because such wireless access points permit free access to all, they rarely utilize encryption, meaning the attacker does not even have to crack an encryption key to view captured traffic as it as been sent in clear text in the first place.

In addition to sniffing traffic from a wireless network in Starbucks, there is little to stop an attacker parking a car outside an organizations head office building and trying to sniff data from the corporate wireless LAN. Do not assume that just because it is a corporate wireless network that it is configured securely; and even if encryption is used, open source key cracking programs like Aircrack-ng[14] can be used to break the key and view the captured traffic. Such risks are not theoretical and actually happen. In May 2007 the wireless network at US clothing retailer Marshalls (the parent company of TJMax) is suspected to have been the cause of a breach of security that resulted in the theft of information on at least 45.6 million credit and debit cards[15].

2. Initial FTP connections (called the "control" connection) between client and server are opened on TCP port 21, and all subsequent command controls are sent through this channel. However, the transfer of data is completed across a separate connection (called the "data connection") that uses a random TCP high port. This data connection process makes firewall support for FTP challenging as is not possibly to clearly identify which ports need to be opened.

   FTP supports two "modes"; Active and Passive. The mode that is used greatly affects the firewall rules required to support the creation of the data connection, and file transfer. In "Active" mode, the client initiates a control connection to the server on port 21 in which it also specifies a data port it is listening on (a randomly selected TCP port >1023). The server will then initiate a new, totally separate data connection back to the client on that specified data port. This can cause problems for

[14] See http://www.aircrack-ng.org/doku.php
[15] See http://www.securityfocus.com/brief/496 for full storey

desktop firewalls (e.g. McAfee Desktop Firewall[16]) installed on the local client because they will see an external host (i.e. the FTP server) trying to initiate an inbound connection to the client on a TCP high port; something that would usually be blocked. This issue can be mitigated by using an application aware firewall on the client. Application aware firewalls inspect the content of packets above the Layer 3 network information (i.e. source and destination IP address and destination port) that non-application aware firewalls stop at. As a result, an application aware firewall would inspect and identify the Layer 4 TCP data that formed the original outbound FTP control connection from the client to the server, note the specified port that the client is listening on for the return inbound data connection from the server, and permit it.

In order to overcome the problem of the server initiating the data connection to the client, passive mode FTP was developed[17]. In passive mode, the client initiates both the command and data connections to the server. In addition to opening the command connection, which it does from a random TCP port >1023 (e.g. 1025) to the server on port 21, the client will also open a data connection to the server, from the specified command connection port +1 (e.g. 1026), to port 20. This eliminates the problem of the server initiating an inbound data connection to the client.

Both Active and Passive FTP create problems for the network administrator because FTP data connections are initiated on random TCP high ports. To permit this traffic through a firewall, the rulebase must be very open (e.g. depending on the mode being used, permit all TCP >1023 to talk to the FTP sever).

As discussed above, application aware firewalls have the ability to inspect the content of an initial ftp command connection and permit the specified data connection port between the server and client for the duration of the data transfer. However, this added functionality itself is known to

[16] See http://www.anidirect.com/products/firewalls/ds_dtfw8.pdf
[17] See http://www.ashton-it.com/cms/index.php?option=com_content&task=view&id=13&Itemid=2

create new security risks. Some firewalls that inspect FTP application layer traffic "may not adequately maintain the state of FTP commands and responses. As a result, an attacker could establish arbitrary TCP connections to FTP servers or clients located behind a vulnerable firewall".[vii] This issue is discussed in more detail in US-CERT (United States Computer Emergency Readiness Team[18]) Vulnerability Note VU#328867[19].

Older firewalls and firewalls that do not support dynamic port opening require an open rule that allows communication between the FTP server and any host on TCP >1023.

So which ftp mode should be used? Active or passive? This decision needs to be made on a case by case basis depending on the software in use throughout the organization and the network architecture. Are client side firewalls being used? Are the network firewalls application aware? Where are the clients and ftp servers located? What operating system is being run on the clients? This last question is important because the built in ftp client that comes with Solaris (a version of UNIX developed by Sun[20],) does not support passive mode (although a 3rd party ftp client such as ncftp[21] could be deployed which would overcome this problem).

The use of random ports makes it is harder for the network administrator to identify and troubleshoot FTP connections because it is not easy to identify what ports they are using. Passive FTP connections are known to have issues with Network Address Translation (NAT) because some NAT devices are not smart enough to change the address inside the FTP packet as well as that of the IP header. In cases where these inside addresses are changed, knock on effects may be created

[18] US_CERT is part of the National Cyber Security Division of the United States Department of Homeland Security and was created to co-ordinate the response to security threats from the Internet. See http://www.cert.org/
[19] See http://www.kb.cert.org/vuls/id/328867
[20] See http://www.sun.com/software/solaris/
[21] See http://www.ncftp.com/

as sequence and acknowledgement fields are changed and not recognized by the client[22].

4. The Secure Alternative to FTP?

Considering the security vulnerabilities WAN's are is susceptible to, accompanied by the weaknesses in the FTP protocol that many organizations rely on to transfer data across them, what alternate solutions are available to security professionals to support secure transfer data across the corporate WAN?

It should be noted that a number of the encryption technologies discussed in this paper are illegal in certain countries throughout the world. For example, "strong encryption", e.g. 3DES, AES, BLOWFISH, cannot be exported to Cuba, Iran, Iraq, Libya, North Korea, Sudan and Syria[23]. Although the current political landscape in these countries makes it unlikely most organizations will have a requirement to implement secure data transfer solutions in them, this fact should still be considered.

FTP over SSH

SSH or "Secure Shell" operates at the network layer (Layer 5) of the 7 layer OSI model and allows data to be transferred between two computers over a secure channel. SSH is currently a proposed standard with the IETF[24] and provides confidentiality and integrity of data through encryption.

SSH allows administrators and users to logon to remote machines through a secure channel and execute commands. This makes it an ideal replacement for older remote management protocols such as Rlogin and telnet, allowing network and system administrators to securely connect to network hosts

[22] These issues are discussed at http://en.wikipedia.org/wiki/File_Transfer_Protocol (see "FTP and NAT devices" section).
[23] See http://www.epic.org/crypto/export_controls/regs_1_00.html
[24] The latest proposal can be found at http://tools.ietf.org/html/rfc4254

such as routers, firewalls, Linux servers, etc for remote management.

SSH also supports the tunneling of existing TCP connections, including FTP. Tunneling is the result of configuring one protocol (e.g. FTP) to send its traffic across another protocols (SSH) connection. This means it is possible to continue using FTP as the underlying technology for file transfers across the network, but use the security benefits provided by SSH to do so securely. When these two protocols are combined, they are generally referred to as SFTP (secure FTP).

SSH is a complex protocol that utilises a number of encryption algorithms, key exchange technologies and features. SSH "provides an encrypted terminal session strongly secured with symmetric encryption algorithms [25]. Encryption is supported by public-key cryptography mechanisms [26] to protect the session key used by a symmetric encryption algorithm[viii]".

SSH was originally created in 1995 by a researcher at a University in Finland after a successful password attack on his campus network. The original version, now called SSH-1, was subject to a number of security flaws which included weak key exchange processes and data poor integrity checking (a number of US-CERT Vulnerability Notes have been released regarding SSH-1 including VU#13877, VU#945216, VU#315308, and VU#25309[27]). SSH-2 was released in 1996 to remediate these security flaws and it is this version that in 2006 formed the basis of the above mentioned IETF proposed standard.

In SSH-1, each host on the network that supports SSH connections has its own unique "private" cryptographic key (this will be automatically generated by the SSH client). In addition, each time an SSH session is initiated by the client to a server, a second key is generated by the server, the

[25] Algorithms for cryptography that use closely related often identical cryptographic keys for both encryption and decryption. This is further discussed in the PGP section.
[26] A form of cryptography where a user has a pair of cryptographic keys; a public key and a private key. The private key is kept secret, while the public key may be widely distributed. This is further discussed in the PGP section.
[27] See http://www.kb.cert.org/vuls/id/13877, http://www.kb.cert.org/vuls/id/945216, http://www.kb.cert.org/vuls/id/315308 and http://www.kb.cert.org/vuls/id/25309

"public" key. When connections are initiated, these keys are exchanged. The client will then validate the server's public key against a locally stored master key for that server. (If the client has not communicated with the server before there will be no local master key to validate against so the client will be asked to either accept or reject a new key sent by the server. It is at this stage of the SSH connection initiation process that is at most risk to man-in-the-middle attacks as there is no way for the client to verify the authenticity of the server sending this key). This process is known as server authentication, i.e. if the validation is successful, this provides identification assurances in that the client knows who it is talking to at the remote end and who the files are being transferred to.

After assuring the identification of the server, a further key is randomly generated by the client and encrypted with both the server's public, and client's private keys. It is this shared "session" key that is used to encrypt and decrypt all data transferred between the two hosts using one of a number of supported encryption algorithms including 3DES[28] and Blowfish[29] (chosen by the client, based on what is supported by the server).

At this stage the client has authenticated the server; it is now necessary for the server to authenticate the client. A number of different methods are available to provide SSH client authentication, the most basic of which is *rhosts*. *rhosts* authentication is implemented by simply placing the client hostname in the *rhosts* file (normally in the */etc* directory); when the connecting clients hostname matches the entry in the servers *rhosts* file, credential free authentication is granted. As noted by R. L. Zeigler in his book "Linux Firewalls", *rhosts* authentication is insecure and should not be used; "RHosts authentication is considered to be a weak authentication mechanism and thus should be disabled o your system".[ix]

In addition to *rhosts*, a number of other methods exists to authentication clients, the most

[28] See http://www.tropsoft.com/strongenc/des3.ht
[29] See http://www.tropsoft.com/strongenc/blowfish.htm

popular of which include public key authentication, host based authentication and password authentication.

Public key authentication is the most secure solution and utilizes a public key cryptography (PKI) system, using both public and private keys. PKI is discussed is greater detail later in this paper but in summary, each client has a private key which only they have access to. A corresponding public key is held by the SSH server. A client will encrypt a message with their private key which the server will decrypt using the corresponding public key. If the message is decrypted successfully, the user is authenticated. PKI requires the server possess a unique public key for every connecting client.

Host based authentication is similar to PKI in that it utilizes public and private keys however, authentication is based on host, no passwords are transmitted. Each clients public authentication key must be installed on the server; all subsequent inbound connection attempts from a client to the server that are encrypted with the clients private key will be permitted, regardless of initiating user. This solution is less secure (as long as you have access to the client machine you have access to the server) but are suitable for use in environments where the server is not trusted and it is not preferable to transmit passwords to it.

A third authentication method uses the standard process of sending a static username and password through the encrypted SSH channel. This method is dependant on the server being configured with a corresponding username and password or being configured to make authentication calls to third party user directories like Microsoft Active Directory (AD). It is possible to encrypt the username and password with the user's client key so that this information is not sent in cleartext. It is also possible to extend SSH authentication with RSA SecureID[30].

SSH-2 was designed to eliminate a number of security vulnerabilities that were identified in

[30] See http://www.calsoftlabs.com/whitepapers/ssh-protocol.html

SSH-1 (man-in-the-middle attacks, IP spoofing) and supports its own internal layered architecture. As well as supporting a larger number of symmetric encryption algorithms (128 bit AES[31], Blowfish, 3DES, CAST128[32], 192 bit AES, 256 bit AES), significant improvements were made to the creation and exchange of the session key by using the Diffie-Hellman[33] cryptosystem key exchange protocol. Integrity checks provided by MD5[34] and SHA-1[35] hash algorithms and user authentication support for Kerberos[36] and x.509 certificates provided further security enhancements.

SFTP works by using a client that will redirect the FTP transfer to a local SSH program, which in turn opens a secure SSH connection to the destination system (data needs to be redirected to the SSH program because FTP and SSH both operate at the same layer of the 7 Layer OSI model). However, because FTP uses more than one TCP connection to transfer data between hosts (one for the session control, one for data transfer), many clients will only tunnel the initial FTP control session. While this provides secure identification, integrity and authentication of the connection (username and password are no longer sent in plain text), the main file transfer, sent over a separate data session, will remain unencrypted plain text FTP. To encrypt the data connection, additional configurations must be supported or the SSH client must have the ability to read, and change, the content of the FTP packets. This issue is mitigated by the fact that many SSH clients provide inbuilt support for SFTP so both the control and data connections are redirected to the SSH session.

Tunneling all traffic through the SSH connection will also make the creation and management of firewall rules significantly easier for the firewall administrator, because the secure channel that SSH creates between two hosts will use just one port; TCP port 22. SSH must be configured on both the

[31] See http://www.tropsoft.com/strongenc/rijndael.htm
[32] See http://javadoc.iaik.tugraz.at/iaik_jce/3.13/iaik/security/cipher/CAST128.html
[33] See http://www.cs.usask.ca/resources/tutorials/csconcepts/1999_1/Table_of_Contents/Public_Key/Diffe-Hellman/diffe-hellman.html
[34] See http://www.tech-faq.com/md5.shtml
[35] See http://www.w3.org/PICS/DSig/SHA1_1_0.html
[36] See http://web.mit.edu/Kerberos/#what_is

server and the client. When doing so, be sure to specify SSH-2 and strong encryption algorithms (e.g. 3DES) are as used because many SSH programs will default to SSH-1 and weaker encryption algorithms (e.g. DES).

While configuration of Windows based SFTP clients is relatively straightforward, configuring the server can be more challenging; especially in a non-Windows environment (http://freshmeat.net/articles/view/1576/ describes the steps involved when configuring a Linux host to act as an SFTP server). FTP over SSH does not address the secure storage of files on network systems when they are not being transmitted, and it is also very easy to deploy insecure implementations of SSH (especially with regards to user authentication as the default configuration on many clients will simply use the client hostname in a local configuration file). SFTP is also rather slow due to the requirement to wait for packet confirmations[37].

Microsoft does not provide an SSH client with any of its standard Operating Systems. As a result, any file transfer solution in a Windows environment that utilizes SSH technologies would require the installation of SSH client software on every host. Such clients are available both commercially (e.g. http://www.globalscape.com/cuteftppro/index.aspx) and as freeware (e.g. http://www.chiark.greenend.org.uk/~sgtatham/putty/). Most Linux, Solaris and MAC Operating Systems come with built in SSH clients.

Secure Copy

To overcome the challenge created by FTP over SSH in instances where only the initial control session is tunneled through the SSH connection and the data transfer is sent in clear text over a separate connection, Secure Copy, or SCP can be used. SCP is another means of securely transferring files

[37] See http://winscp.net/eng/docs/protocols

between two remote hosts, using the SSH protocol.

SCP is an older protocol and like FTP, does not support data encryption. In fact, SCP does not even support authentication, it relies on SSH for this functionality. The main advantages of SCP is that it requires only one connection for all data (hence everything is transferred through the secure connection), and is different to FTP in that it can maintain date and timestamp attributes.

Unfortunately, support for SCP is limited to Linux and UNIX based systems and does not come with a graphical front end, instead relying on the CLI. This makes a deployment across a wide user base challenging. SCP is efficient at transferring data but does not allow data transfers to be interrupted. This can be a significant problem when transferring files over a WAN. As a result, SCP "is often a more suitable choice when setting up unattended file transfers using scripts".[x]

A table that clearly identifies the main differences between SFTP, FTP and SCP is available at the following link: http://winscp.net/eng/docs/protocols#protocol_comparison.

FTP over SSL/TLS

FTP over SSL, commonly known as FTPS is another means of using FTP to securely transfer files between two hosts. FTPS uses the cryptographic protocol Transport Layer Security[38] (TLS[39]) to encrypt data during file transfer and provide secure authentication. Securing FTP with TLS is a proposed standardized in RFC 4217 by the IETF[40].

TLS uses cryptography to provide endpoint authentication and communications privacy by encrypting all data transmitted between two hosts. The protocol that predominantly uses TLS is

[38] Although the IETF superseded SSL with TLS, the SSL name has gained such notoriety that people still refer to the protocol as SSL or SSL/TLS
[39] See http://tools.ietf.org/html/rfc4346
[40] See http://tools.ietf.org/html/rfc4217

HTTP, better known as HTTPS, to secure Internet web pages. VPN technologies have also taken advantage of TLS as a means of providing clientless VPN access from mobile user laptops back to the corporate network over the Internet (often referred to as SSL VPN's). However, TLS can also be used to secure FTP data that is transmitted between two hosts across a WAN.

Before any data is transmitted, a handshake procedure takes place between the client and server to agree which encryption algorithm will be used to encrypt the data (e.g. 3DES, AES; the strongest algorithm that both parties support is generally chosen), and also which hash functions will be used to guarantee the integrity of the data. Hash functions create a digital fingerprint (or message digest) of the data being transferred by executing a small mathematical computation against the data and outputting a unique, fixed-size string. This string is added to the data being sent. Upon receipt of the transmitted data, the receiving host will re-run the same mathematical computation and check it against the digital fingerprint included with the data. If the two match, it is safe to assume the data has integrity, and has not been changed. If the two are different, there is no integrity. Common hash functions include MD5 and SHA-1.

TLS differs to most other protocols in that it does not operate at just one layer of the 7 Layer OSI model; it is neither an Application nor Network layer protocol. As a result, it is said to "sit" between the two layers. This provides TLS with unique flexibility that allows it to encrypt a single application (or protocol) as opposed to a whole group of applications[41]. This means that when encrypting FTP all traffic is securely encapsulated without the need to have a client route it through a separate connection, as required when using FTP over SSH.

FTP over SSL is reasonably easy to implement as most FTP Servers, and more recently clients, support it. CoreFTP[42], one of many free Windows FTP clients (Pro and corporate versions available

[41] See http://searchsecurity.techtarget.com/expert/KnowledgebaseAnswer/0,289625,sid14_gci1266329,00.html
[42] See http://www.coreftp.com/

for purchase) supports SSL and TLS authentication and data transfer. Such FTP clients would need to be deployed to every host requiring access to the FTPS server.

When configuring FTPS at the server end, attention needs to be given to the installation of an SSL certificate. Although FTPS will work using a default SSL certificate, if the server is to be accessed over the Internet, an SSL certificate issued by a Certificate Authority[43] (e.g. Verisign[44] or Entrust[45]) should be installed to provide guarantees to users that they are communicating with the correct server. This can be a time consuming task, especially if you do not have an existing account with a CA. New registration processes[46] can take a number of weeks to complete as verification and security checks are completed on the purchaser, by the CA). Additional detail on SSL certificates is provided later in this paper (see WebDAV over SSL).

Microsoft plans to include improved builtin FTP services in the forthcoming release of Windows Server 2008, one of which includes an FTP user interface that can be configured to use SSL (existing Microsoft Widows server products do not provide built support for FTPS; 3rd party software is required). There will be a number of pre-requisites required to use these features, most of which include running the latest software from Microsoft (Windows server 2008, IIS 7.0[47]).

Similar to SFTP, FTPS does not address the secure storage of files on network systems when they are not being transmitted.

WebDAV over SSL

Web-based Distributed Authoring and Versioning (WebDAV) was developed by an Internet

---

[43] A trusted third party that issue digital certificates containing a public key to verify the identify of the owner
[44] See https://www.verisign.com/
[45] See http://www.entrust.com/
[46] See http://www.verisign.com/SSL/SSL-information-center/SSL-enrollment-process/index.html for Verisign process
[47] See http://www.iis.net/articles/view.aspx/IIS7/Managing-IIS7/Using-FTP-Server-in-IIS7/Using-FTP-over-SSL.

Engineering Task Force (IETF) working group with the goal of defining standard functionality that would allow users to easily create, change and move documents across remote servers; typically servers that are part of the world wide web (www). It was originally defined in RFC #2518[48]. A WebDAV community resource site is also hosted at http://www.webdav.org/.

The www (or "web") is generally a read only medium, providing millions of user's access to a vast array of documents, linked together with "hypertext", and accessed over the Internet. In addition to read access, early pioneers of the web originally wanted users to have both editing and write permissions to the documents hosted on it but, as it grew and evolved, read only access became the default. This was mainly because Hypertext Transfer Protocol (HTTP[49]), the communications protocol that was (and is) by far the most prominent in use on the web and forms the basis of the technology used to transfer documents across it, did not support these editing and write requirements. The IETF working groups goal was to change this and WebDAV, a new group of functions (or "methods") that were added to the HTTP protocol, was the way they achieved it.

The newly created WebDAV methods that were added to HTTP (to become known as HTTP version 1.1), allow users to work with remote files as if they were local resources. Users are able to easily create new directories on remote servers, to edit documents (including the ability to "lock" them beforehand, and "unlock" them afterwards), to change or delete the properties or permissions associated with documents, and most importantly as far as data transfer functionality is concerned, to move or copy documents from one location to another.

A web server is required to provide access to a WebDAV directory. WebDAV is supported by all mainstream web servers including Apache 2.0[50] and Microsoft Internet Information Services (IIS[51]).

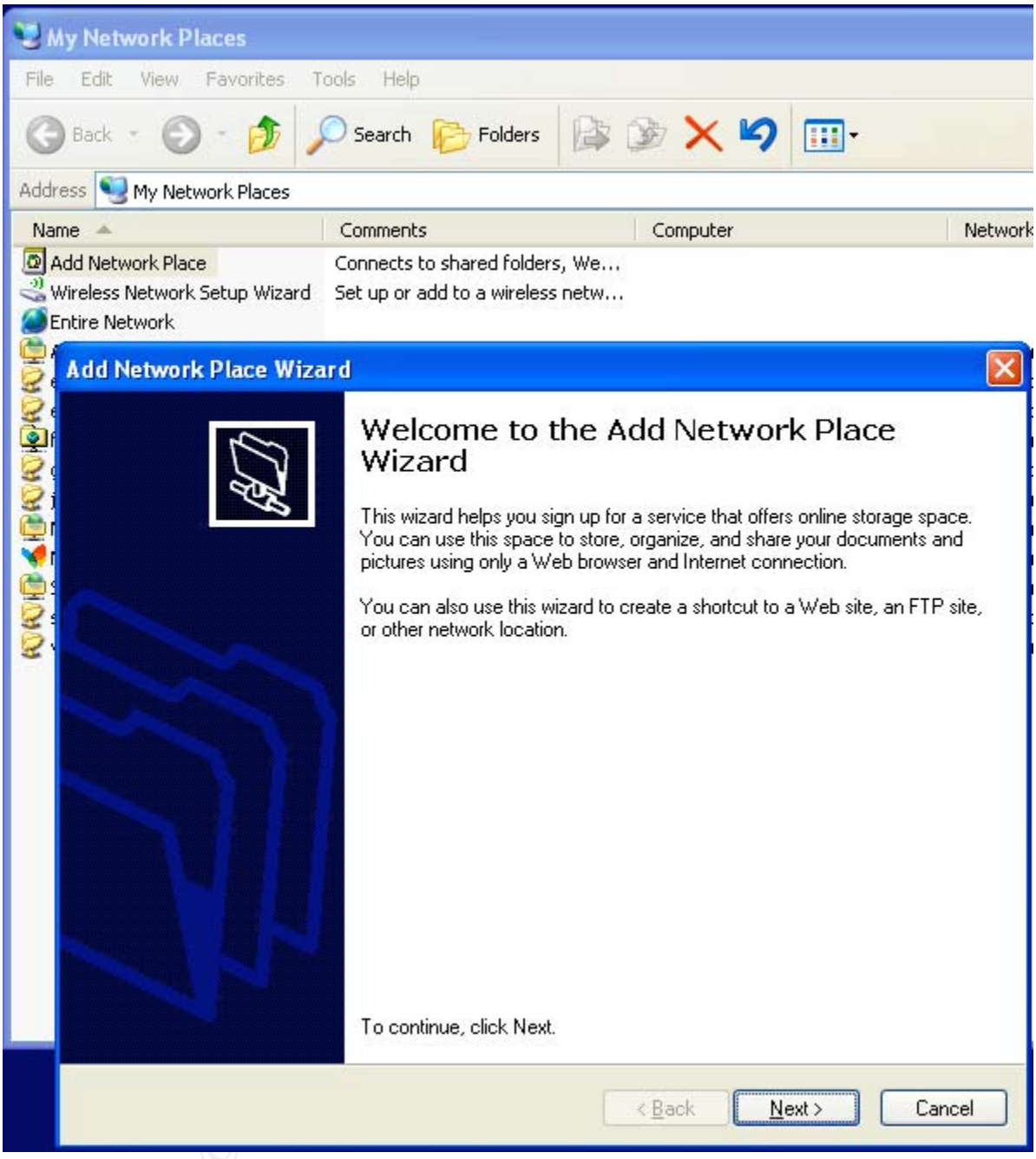
---

[48] See http://tools.ietf.org/html/rfc2518
[49] See http://tools.ietf.org/html/rfc2068
[50] See http://httpd.apache.org/download.cgi
[51] See http://www.iis.net/default.aspx?tabid=1

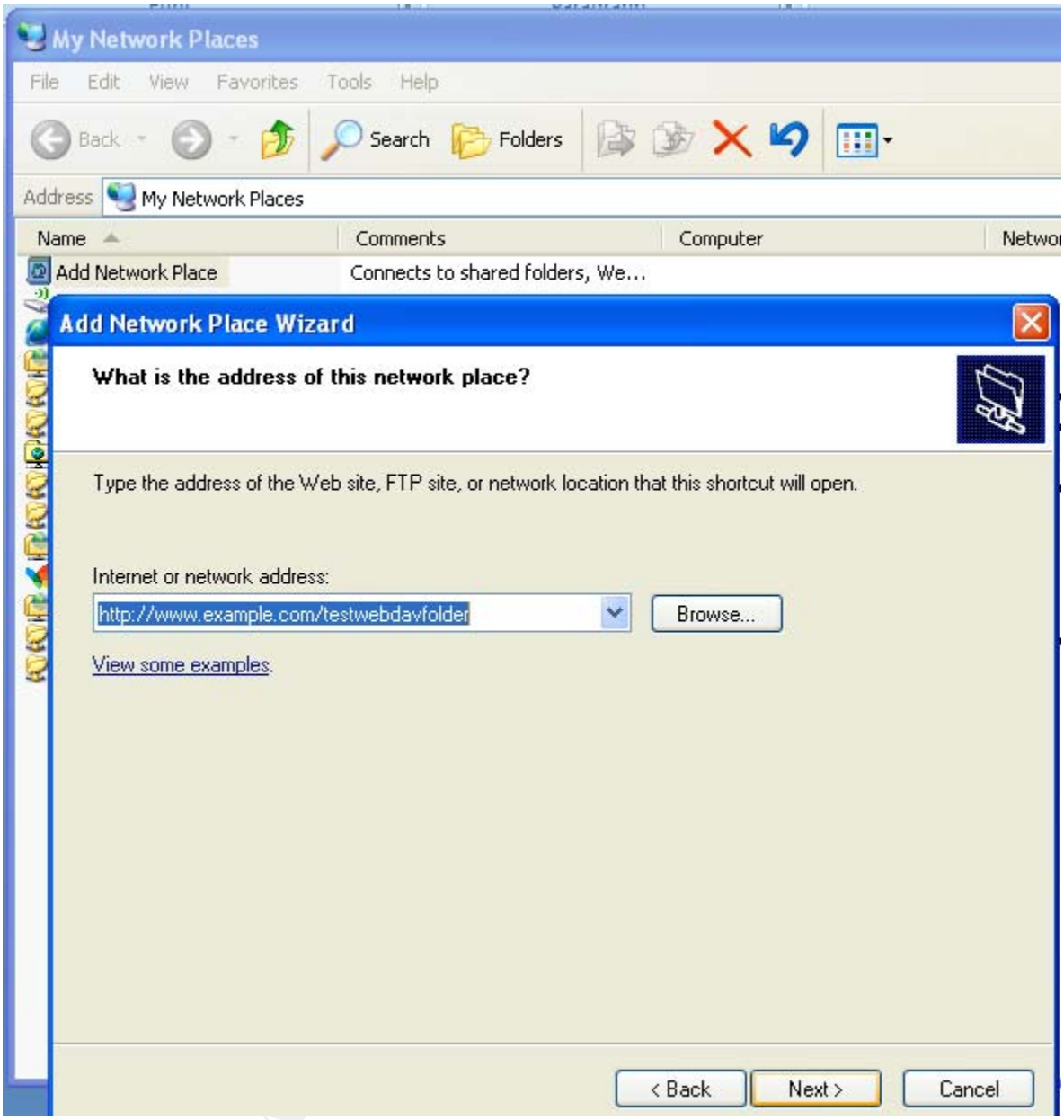
As specified in documentation published on the Microsoft website, WebDAV is very easy to set up; "Setting up a WebDAV publishing directory on your server is as straightforward as setting up a virtual directory through IIS Manager[xi]". To add to this reported easy setup, there are various ways a client can access a WebDAV publishing directory (many of which are wizard based); through a basic Windows Client (make a connection in "My Network Places"), a Web Browser (e.g. Internet Explorer versions 5, 6 and 7, Firefox, Mozilla etc, by entering the path to the remote folder in the URL field) or by using inbuilt functionality in Microsoft Applications such as Office and XP. The below screen shots detail the steps to follow to map to a WebDAV share on Windows XP SP2 through My Network Places:

1. Open "My Network Places" by clicking desktop shortcut icon

2. Double click "Add Network Place" to launch the "Add Network Place Wizard"



3. Follow the wizard setup until you are asked for the address or "path" to the WebDAV share. Enter it in the provided field and select "Next" (format will normally be http://www.example.com/testwebdavfolder, this is specified when the administrator creates the share).



4. If authentication is required, at this time you will be prompted to enter user credentials

5. The WebDAV share will appear as a mapped drive under "My Computer"

Despite WebDAV's apparent ease of configuration and use, many users report slow file transfer speeds and question reliability, especially when accessing directories via a browser. The

following bullet points contain text that has been extracted from three of the hundreds of postings available on the Internet (search for "webdav slow") that discuss WebDAV being slow. It should be noted that these postings does not account for local configuration or implementation errors, inadequate hardware specifications etc. or the equally high number of potential successful implementations;

- "WebDav file transfer is way too slow to be very useful--to be used on a daily basis"[xii]
- "Opening files located in your space when connected using WebDAV is very slow and may require that you re-login multiple times".[xiii]
- "I've got WebDAV set up for a site and I'm testing the access using Dreamweaver and the Finder. I'm finding that WebDAV is really slow to connect with both Dreamweaver and the Finder."

When configuring WebDAV, because it is hosted on a web server it is important to pay close attention to the security configuration of that web server. Products like Microsoft IIS version 5.0 are configured by default with insecure settings and are affected by numerous known security vulnerabilities[52].

Microsoft IIS is not the only web server to suffer from known security vulnerabilities. Apache has similar issues[53]. Of course, it is not only the web server that needs to be hardened; as is the case with any file transfer solution, the host OS must be hardened, patched and closely monitored (e.g. log review) to prevent unnecessary access and successful malicious attacks. File and directory permissions must also be applied to shares to restrict access to only authorized users.

---

[52] http://www.cit.cornell.edu/security/seminars-past/iis-july01/ and http://www.sans.org/reading_room/whitepapers/webservers/304.php are two of a large number of articles available online written to help system administrators configure IIS 5.0 securely. They highlight the insecure default settings of IIS 5.0, vulnerabilities affecting it, and known exploits of these vulnerabilities.

[53] See http://httpd.apache.org/security/vulnerabilities_13.html

WebDAV is able to support multiple transfers through a single connection. This reduces much of the "control" connection handshake overhead and commands seen with FTP. However, WebDAV does not support all file transfer needs discussed in this paper because HTTP transmits data in clear text. This means that deployed on its own, WebDAV it is not a suitable solution.

Secure Sockets Layer (SSL) was designed to secure unencrypted web sessions that use HTTP, i.e. it supports the creation of an encrypted link between a web server and a browser. SSL on its own provides very little; it is effectively a handshake and encryption protocol, but, when bundled with HTTP (or, in fact, any TCP based protocol), it provides a secure mechanism under which the protocol can operate; commonly known as HTTPS.

Therefore, it is possible use the file transfer technologies that WebDAV supports through HTTP, and secure them with SSL (commonly referred to as HTTPS). One advantage of WebDAV is that everyone has an HTTP enabled web browser. This means that to use WebDAV functionality to move files across the network, it is not necessary to deploy local client software to any client machines. And, because HTTPS uses port 443 which is commonly an open port on many internal and external facing corporate firewalls, implementation is that much easier as fewer changes are required to the network architecture.

In addition to hardening the web server and base OS, in order to use WebDAV over SSL, there is a requirement to apply an SSL certificate to the web server. SSL certificates allow clients to communicate securely over SSL as they confirm that the server hosting a file is indeed the "real" server and not a spoofed version (i.e. confirms the validity of the server), and also provides the key to encrypting and decrypting data, allowing it to be being securely transported through the SSL tunnel. Further information regarding the different types of SSL certificates available and how to apply then to a web server, can be found at http://www.psoft.net/HSdocumentation/user/SSL.html.

When configuring a web server to support file transfer using WebDAV, a best practice means of

doing this usually involves configuring two separate web sites; one to support "read-only" traffic and one to support WebDAV functionality. This adds additional configuration, support and management to this solution.

From Windows 98 onwards, Microsoft introduced extended functionality to Windows Explorer that allows WebDAV shares to be assigned drive letters and subsequently accessed by various applications running on the local client. This functionality was originally called "web folders" but from Window XP onwards, became known as WebDAV mini-redirector. Whilst providing useful functionality, all versions of the redirector software, including that supported by Windows Vista, have confirmed security issues[xiv].

FTP layered on top of IPsec

Encryption based protocols such as SSL, TLS and SSH operate at the transport layer (Layer 4) and up of the OSI 7 layer model. Whilst this provides encryption with low overhead (most of what is encrypted is actual "data" from the application, not bit fields from lower level protocols that are added to the packet when it is encapsulated), such solutions have limitations because they are unable to support all TCP and UDP based protocols that operate at these layers.

Encryption is often a requirement for network and security administrators because many of the protocols in use on corporate networks today (e.g. TCP, UDP, HTTP, telnet), do not provide security functionality. This problem is not limited to Layer 4 and higher protocols; it is also the case for the Internet Protocol (IP) which operates at Layer 3, and is the primary reason for the creation of a suite of encryption protocols commonly referred to as IPsec (IP Security)[54].

[54] A full list of goals, milestones and rfc drafts developed by the original IPsec IETF Working Group can be found at http://www.ietf.org/html.charters/OLD/ipsec-charter.html. One of the high level goals of the working group was to

Internet Protocol security (IPsec) is a set of open standards that can be used to secure communications over IP networks through the use of cryptographic algorithms and protocols. As documented by Microsoft in the TechNet article 'What is IPSEC' (2003), "IPsec supports network-level data integrity, data confidentiality, data origin authentication, and replay protection." [xv] IPsec operates at the network layer (Layer 3) of the OSI 7 Layer model providing encryption on a per packet basis. As a result, applications, users and services do not need to be IPsec "aware". IPsec encryption is a transparent operation that occurs without the need for code and program changes, or user intervention.

IPsec is not, in itself, an encryption program or algorithm like PGP or SSH. It is a group of security technologies that when combined, provides security to the Internet Protocol. There are no rules stating which of these technologies need to be used, or in what combination, that decision is left to the network administrator.

IPsec protocols, as defined by RFC's 1825–1829[55], are built into IPv6[56] and mandatory. However, the requirement of network administrators to secure existing IPv4 implementations meant that IPsec support was retrospectively added as an optional feature[57]. IPsec encrypted packets can be routed and switched across any IP network (although there are some caveats when using NAT, see later), to any IP client, without the need for additional hardware or software upgrades or modifications. As a result, almost everyone is able to support it.

Some of the most popular technologies that IPsec supports include Authentication Header (AH, see RFC #2404[58]) which guarantees data integrity and origin, Encapsulating Security Payload (ESP, see

---

"develop mechanisms to protect client protocols of IP".

[55] See http://tools.ietf.org/html/
[56] See http://www.ipv6.org/
[57] See http://csrc.nist.gov/archive/ipsec/index.html#papers
[58] See http://www.faqs.org/rfcs/rfc2404.html

RFC #2406[59]), for data origin authenticity, integrity, and confidentiality, DES and 3DES for encryption of data, and MD5 and SHA for hashing functions. Internet Key Exchange, (IKE, see RFC #2409[60]) is used to negotiate the parameters of secure communication between two IPsec peers (e.g. which encryption algorithm and hash functions to use, inactivity timeouts, key renegotiation requirements etc). These parameters form what is known as the Security Association (SA, see RFC #1825[61]) between the peers. IKE operates on port UDP 500 and is key to IPsec functionality.

IPsec has two modes of operation, transport and tunnel. In transport mode, only the payload of the packet is encrypted; the IP header information is left in clear text. Transport mode is most commonly used on internal networks to create secure peer-to-peer communications channels between a small number of critical hosts (e.g. domain controllers and global catalogue servers). If packets are sniffed, the IP header information is visible but the critical data remains secured. In all but the most secure environments (DOD etc.), it makes little sense to place additional overhead on the encryption process by encrypting header information in addition to the packet payload, if the traffic is destined to stay internal of the network boundary. This is because an attacker who can sniff IPsec encrypted packets will also be able to sniff unencrypted packets; providing him easy access to IP address schemes and other network topology information regardless.

In tunnel mode, the whole IP packet is encrypted and subsequently encapsulated into a new packet for routing across the network. As a result, tunnel mode is often used to create point-to-point IPsec "tunnels" between remote office locations to form private WAN's across the public Internet. Because most internal corporate networks are configured with private non-Internet routable IP addresses (as defined in RFC #1918[62]), tunnel mode is required to route data across the Internet because the original IP packet is fully encapsulated and assigned the public IP address information of the tunnel

[59] See http://www.faqs.org/rfcs/rfc2406.html
[60] See http://www.faqs.org/rfcs/rfc2409.html
[61] See http://www.faqs.org/rfcs/rfc1825.html
[62] See http://www.faqs.org/rfcs/rfc1918.html

endpoints (most commonly a router or firewall but can be any device that supports the IPsec protocols). Details about the internal network topology are also protected should the packets fall into attacker hands[63].

In transport mode, because the packet is encrypted and decrypted by the client at either end, data is not sent over any part of the network in clear text. This is slightly different than in tunnel mode. Encryption in tunnel model is normally done by an IPsec gateway such as a firewall, router or dedicated VPN appliance. This means that while packets sent between two gateway devices will always be encrypted, data will be sent in clear text format over the local network from the client to the IPsec gateway. As stated above, in all but the most secure environments (e.g. DOD) this should not be an issue however, it is a point to note.

Because tunnel mode encrypts the whole IP packet and adds new IP address fields, the new encapsulated packet is at least 28 bytes larger than the original (this varies significantly depending on which IPsec protocols are used. Cisco field notice advisory 62394[64] is an example of how confusing this can be). In addition to creating more data to transfer across the WAN, IPsec encapsulation often creates the requirement to fragment (or split in two) the original IP packet because these additional 28 bytes (or more) take new the packet over the IP packet size limit (see RFC #891[65] for full IP specification). Transport mode only encrypts the packet payload, adding at least 8 bytes to the size of the packet. This makes it more efficient that tunnel mode and reduces the requirement for fragmentation. IP packet fragmentation is a very broad subject that will not be discussed in any more detail in this paper[66].

If there is a requirement to use AH to provide a hash value of the IP header (i.e. to secure the

---

[63] Further reading on IPsec tunnel mode is available at http://technet2.microsoft.com/windowsserver/en/library/12eb6a6f-25cb-4af4-a659-59d9ff8de3361033.mspx?mfr=true.
[64] See http://www.cisco.com/en/US/products/sw/iosswrel/ps5207/products_field_notice09186a0080697964.shtml
[65] See http://tools.ietf.org/html/rfc791
[66] For more information on IP fragmentation, and the many factors that affect it, see http://www.cisco.com/en/US/tech/tk827/tk369/technologies_white_paper09186a00800d6979.shtml

integrity of the packet), while IPsec transport mode will support this, a knock on effect means that the use of Network Address Translation (NAT) becomes impossible because such packet "interference" would invalidate the IP header hash value. ESP does not have a problem with NAT.

IPsec technologies can be used to provide data authentication and integrity as well as encryption, or just data authentication and integrity with no encryption, and visa versa. The primary goal of the technology is to be flexible and versatile. However, while this has been achieved, with flexibility and versatility comes confusion, and although IPsec is based on IETF standards, implementation and support issues have created challenges for organizations trying to implement it.

IPsec supports communication with any destination through security policy settings. Microsoft support for IPsec means it can be configured via a GUI and although configuration can be complex, tools like the IPsec Configuration Tool[67] are available to make life easier. IPsec can be very complex to configure on systems that are primarily configured via the CLI, such as Cisco routers[68]. This process can only be considered straight forward to the most experienced network engineer.

In addition to placing increased overhead on network routers and bandwidth as a result of fragmentation, IPsec also places additional overhead (and subsequent delay) on other participating hosts on the network. The negotiation process to create and maintain SA's between hosts, the encryption and decryption of data at Layer 3 and the encapsulation and decapsulation of IP packets all contribute. However, this overhead and delay is difficult to quantify because it is dependant on processor speeds and network bandwidth availability. Dedicated IPsec encryption/decryption network interface cards (NIC's) and modules are available for most servers and network devices, which transfer this load off the main processor to all but eliminate this overhead.

---

[67] See http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=7D40460C-A069-412E-A015-A2AB904B7361

[68] See following link to Cisco website that details how to configure a virtual tunnel interface with IPsec. http://www.cisco.com/en/US/tech/tk583/tk372/technologies_white_paper0900aecd8029d629.shtml.

Firewall configuration is straightforward when IPsec is used; the ESP and AH protocols (50 and 51) operate at the same layer as TCP and UDP so are easy to configure and permit in the rulebase. Other ports associated with IPsec services, most commonly those used during the SA negotiation phase (e.g. IKE uses UDP port 500) would also need to be permitted.

Both modes of IPsec can be configured to encrypt all traffic, all traffic originating from or destined to a specific IP address, only traffic on a specific port, or a combination of each. If using FTP to transport data across the network, it would be necessary to encrypt packets based on destination IP address because the exact port number that will be used for the FTP data connection would be unknown. In tunnel mode, data would only be encrypted during transmission between the network tunnel endpoints (e.g. the network routers or firewalls). Traffic between the client and source VPN gateway, and destination VPN gateway and FTP server, would be sent in clear text format.

In addition to the above, both tunnel and transport modes of IPsec would only provide encryption during the transportation of data. Data would be stored on both the source client and destination server in clear text, unencrypted format.

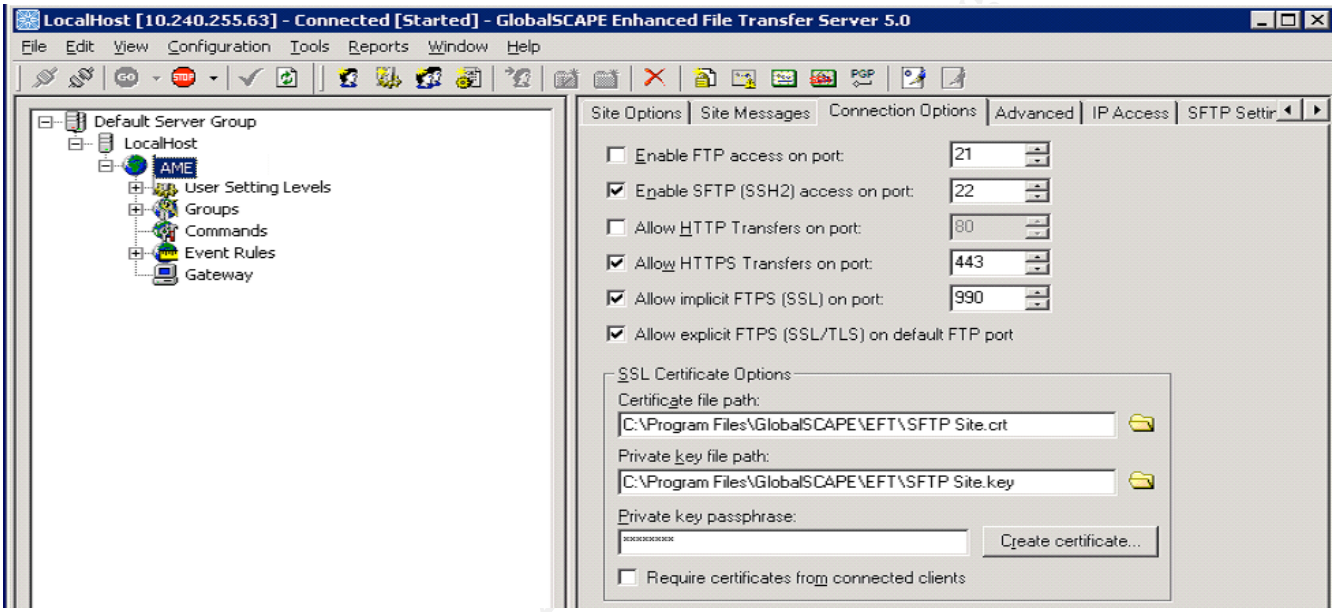
"Off the Shelf" data transfer tools

As the demand for organizations to securely transfer data over the WAN has grown, so has the number of "off the shelf" vendor products that are available to meet this demand. Such products provide support for all of the data transfer solutions and encryption protocols discussed so far this paper (FTP, SFTP, FTPS, WebDAV, DES, 3DES, etc), and are configurable via user friendly GUI.

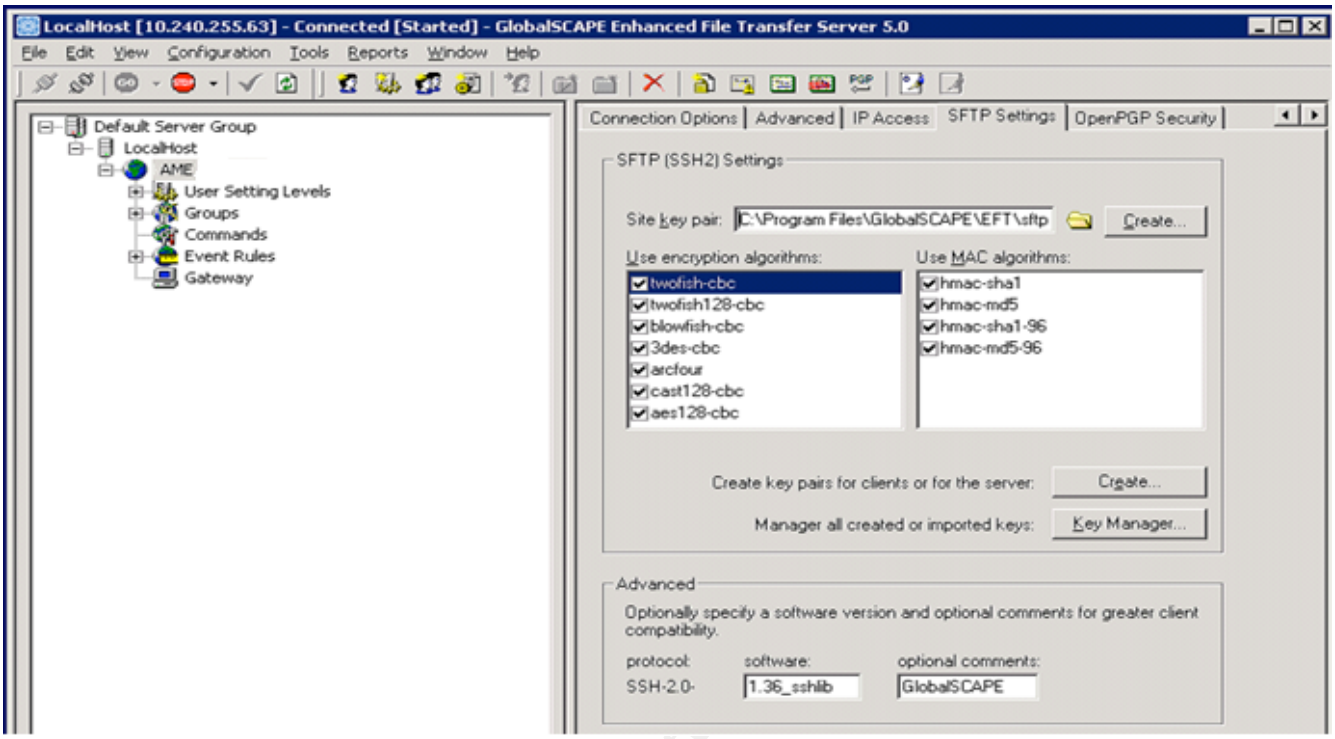
One such product is GlobalSCAPE's EFT (Enhanced File Transfer[69]). EFT provides support for the above listed protocols (and others), can be integrated with AD for access control and

[69] See http://www.globalscape.com/eft/

authentication, can be used for secure file transfer over the internal corporate WAN or the Internet (through the implementation of a proxy "DMZ Gateway") and has extensive auditing, logging and reporting capabilities. In addition to providing this vast range of functionality, such solutions are remarkably easy to install and configure. Below is a screen print that shows the configuration console of GlobalSCAPE's EFT server 5.0.



Configuring this product is literally as easy clicking a box to identify which protocol you wish to enable over which port (e.g. SFTP over port 22), and then clicking on the appropriate tab to define the parameters of such connections. Below is a screen print of the SFTP configuration tab that is accessed from the above screen. This is where the system administrator specifies which encryption algorithms and hash functions are to be used for inbound connections, as well creating, importing and managing private keys. Additional functionality includes the ability to specify a minimum client version that must be deployed to all hosts initiating inbound connections.



Unfortunately, products like EFT can be expensive to purchase and will only run on Windows based systems. Client software must also be deployed to all hosts. In addition, it is important to note that ease of installation and configuration does not eliminate the requirement to fully understand the technology and protocols being used. EFT will support clear text data transfers using ftp and also SSH-1. As result, it is just as easy to deploy an insecurely configured off the shelf GUI driven data transfer solution as it is to deploy a securely configured one.

Secure Alternatives to Unencrypted Data

The technologies discussed in this paper to date have only addressed the encryption of data during transmission over the network. They do not account for the security of data when it is at rest, i.e. stored on a company desktop PC or more importantly, a users laptop that is frequently taken out of the office environment and lost or stolen in airports, taxis, trains, busses or even client sites. In fact, it would appear that the average corporate laptop may be most at risk from theft when it is actually in the office. A survey of stolen laptops conducted in October 2005 by data security provider Credant Technologies[70] found that "The office is the most common place laptops are stolen, tallying 29 percent of all laptop disappearances worldwide". (While it is noted that this means 71% of laptops are lost or stolen outside of the office, 29% is a significant number). Credant CEO Bob Heard commented "what we discovered were corporate environments that are careless and even reckless with laptops, many of which contain crucial company and personal data".

As of June 2007, 35 US States had laws on their statute that place notification obligations on companies to publicly report data security breaches[xvi]. These laws include the loss of laptops containing sensitive client data. As a result, there has been a surge of high profile public cases that disclose the theft and loss of corporate laptops[71], most of which are heavily reported within the media and openly discussed on Internet blogs[72].

When a laptop is lost or stolen the value of the data to the organization nearly always

[70] See http://www.credant.com/content/view/101/105/
[71] One of the most widely reported and publicized events to have occurred to date was the loss of a laptop containing the personal information of more than 26.5 million veterans from the Department of Veterans Affairs in May 2006. See http://www.news.com/Veterans-data-swiped-in-theft/2100-1029_3-6075212.html
[72] One of many blogs is the InfoSec blog available at http://infosecblog.antonaylward.com/2007/01/01/2006-the-year-of-the-laptop-stolen-that-is/

outweighs the value of the hardware[73]. In addition, organizations must also consider the overall cost to the company in damaged image, reputation and future lost business; a figure that is almost impossible to calculate. However lost laptops are not the only threat to an organization if data is stored at rest in unencrypted format. If users are able to freely copy unencrypted data from hard drives onto removable media like USB thumb drives, the organization is putting itself at great risk of corporate espionage or the aggrieved staff member who wishes to carry out a vendetta against his or her employer.

It is, and has been for some time, generally accepted by most IT security professionals that insider threats to corporate data are as serious as external threats[xvii]. Employees with a vendetta against the company, staff that plan to leave the company and steal sensitive data before doing so, users who may be paid to harvest firm secrets in acts of corporate espionage, and, the most prevalent, the normal employee who has a poor attitude to information security and the data at his fingertips, all pose a significant risk to corporate data. The abundant and cheap availability of USB thumb drives, small portable mass storage devices and read/write CD drives has done little to mitigate this threat.

It is crucial that organizations implement solutions that secure data while it is at rest. Solutions and technologies that support such requirements include:

PGP

PGP and PKI Background

Pretty Good Privacy, or PGP as it is commonly referred, is a group of software applications first developed by Phil Zimmerman[74] in 1991 to provide cryptographic encryption and authentication. PGP was originally designed to provide email encryption (for both content and attachments) but has

[73] The article posted at http://www.news.com/Could-your-laptop-be-worth-millions/2100-1029_3-6032177.html suggests "the average laptop could contain data worth almost $1 million".
[74] See http://www.philzimmermann.com/EN/background/index.html

since developed into a mature suite of products that provide full disk encryption, network server file and share protection, digital signatures and instant messaging protection, all of which can be deployed, managed and monitored from a centralized network console. The underlying technology of the pgp tools developed by Phil Zimmerman form the basis of the OpenPGP Internet standard (as detailed in the IETF proposed standard #2440[75]). This means that while the underlying technology of PGP is free, it is possible to purchase products that utilize PGP technologies from the company "PGP Corporation" as well as a number of other 3rd party vendors.

PGP uses both synchronous (also known as private key cryptography, or symmetric encryption) and asynchronous (also known as public key cryptography, or asymmetric encryption) key encryption algorithms. Synchronous encryption algorithms supported by PGP include AES, 3DES, and IDEA[76] use the same key (often referred to as a shared key) to encrypt and decrypt a message. Asynchronous encryption algorithms uses two keys; a public key which is available to everyone, and a private key which is only available to the person decrypting the message. Asynchronous encryption algorithms require key exchange protocols[77]; PGP supports Diffie-Hellman, DSA[78], and RSA[79].

Encrypting and decrypting a message using symmetric cryptography is relatively straight forward. Two parties agree to use an identical key (normally just a shared password); the sender runs the message through a PGP supported encryption algorithm using this key, and sends the output to the receiver. Upon receipt, the receiver runs the encrypted file through the same encryption algorithm, using the same key, to decrypt the message.

Symmetric encryption is popular because of its ease of use. However, this can also be its

---

[75] See http://tools.ietf.org/html/rfc2440
[76] See http://www.finecrypt.net/idea.html
[77] See http://www.bioid.com/sdk/docs/Cryptography.htm
[78] See http://home.pacbell.net/tpanero/crypto/dsa.html
[79] See http://www.di-mgt.com.au/rsa_alg.html

downfall. Users at both ends need to securely store the encryption key and ensure secure channels of communication when originally agreeing a key to use. Because only one key is used to encrypt and decrypt all data, should that key be compromised, all future transmissions may also be compromised. Keys should also be changed on a periodic basis (depending on sensitivity of data, risk of key being exposed and amount of use).

To encrypt and decrypt a file using asymmetric cryptography, each party requires two keys; a public key and a private key. These two keys are mathematically linked[80] to each other and are created when a randomly generated number is fed into a key generation function (this would be a piece of software included in the PGP application).

Asymmetric cryptography functions as follows. Two parties, A and B, exchange public keys but keep their private keys to themselves. A and B must now agree which encryption algorithm to use to encrypt their file. An encryption algorithm is a mathematical function that takes an input of plain text (i.e. the file to encrypt) and the user generated public or private key, and outputs' ciphertext, essentially the encrypted file that is unreadable. When the file arrives at the other end and is ready to be decrypted, the same encryption algorithm (or mathematical function) is used to decrypt the file using the corresponding public or private key that was used to encrypt the data. The encryption algorithm is not a secret; the way it works is a published standard. It is the key that is the secret and holds the power to decrypting the file.

So, if A is to send a message to B and ensure its confidentiality, the text would be encrypted using B's public key. When B receives the encrypted message, it will be decrypted, as only B can, with B's private key. Providing B has properly maintained the secrecy his private key, no one else will be able to decrypt A's message. But how does B know it was A that sent the message in the first place?

---

[80] Detailed mathematical reading available at http://www.muppetlabs.com/~breadbox/txt/rsa.html

There is no authenticity associated with this message because B has distributed his public key to numerous people; anyone could have encrypted that text. What if A were to encrypt the text with his private key? Upon receipt of the encrypted text, B would decrypt the message using A's public key and know it was sent by A because only he has a copy of his private key. But A has also distributed his public key to other parties which means they also have the ability to decrypt the message. We gain authenticity but loose confidentiality.

A message encrypted with a public key can only be decrypted by the holder of the corresponding private key. This ensures confidentiality. A message encrypted with a private key can only be decrypted with the corresponding public key. This ensures authenticity (i.e. it is possible to verify who sent it; also known as non-repudiation, meaning the sender cannot deny sending it). However, is it possible to achieve both confidentiality and authenticity? What if A were to encrypt a file with B's public key and then append a hash of that file which is encrypted with A's private key? If B is able to decrypt the hash with A's public key, B knows that A sent it. Provided B has kept his/her private key secure, only B will be able to decrypt the file (i.e. using that private key), thus providing confidentiality. In effect, this creates a unique shared key that can be used in a symmetric cipher that will provide both confidentiality and authenticity[81].

Because possession of keys is such an integral part of PGP, key distribution is of huge importance. When A encrypts a message using B's public key, how can A be sure the key he is using belongs to B? Although PGP has built in capabilities to prevent key tampering, this does not guarantee that the public key in B's certificate actually belongs to B. If there is no trust in the way encryption keys are managed and distributed, how is it possible to trust the technology?

PGP offers two solutions to key distribution. The easiest (and cheapest) is known as the 'web

[81] A full description of this process, in addition to a detailed overview of PGP, can be found at http://www.pgpi.org/doc/pgpintro/

of trust'[82]. The web of trust concept was also created by Phil Zimmerman and effectively makes everyone a certificate authority. Any user can validate and trust another user's public key, however, that trust is only valid to another user if they trust you (or as Zimmerman states, you are a "trusted introducer"). For example, if A receives B's public key securely and fully trusts that key, he can validate it. This validation is placed within A's public key. All people that receive A's public key and trust A as a trusted introducer will now also trust B's public key. Anyone that does not trust A as a trusted introducer will not trust B's public key. As a result, a "web of trust" is created.

The second key distribution solution that PGP supports is known as PKI (public key infrastructure) and relies on a third party Certificate Authority (CA) to create, sign and validate the authenticity of keys so you do not have to. Using the CA's public key, anyone wanting to verify a certificate's authenticity verifies the issuing CA's digital signature, and hence, the integrity of the contents of the certificate. However, you have to explicitly trust all decisions made by the CA.

There are a number of factors that need to be considered when choosing a key distribution technology. These include budget, the number of users in the organization and their level of competence, and the predicted level of PGP use. A PKI is certainly the easiest solution from the users perspective (they do not make any decisions, the CA does that for them), but is without doubt the more challenging solution for the administrator. As noted by Luther Martin from Voltage Security[83], an enterprise security company focusing on the information encryption market, "public-key infrastructure (PKI) ….. is often expensive and difficult to implement and use".[xviii] In December 2003 the US Government Accounting Office published a paper titled "Status of Federal Public Key Infrastructure Activities at Major Federal Agencies and Departments," (GAO-04-157) where is said "One study by the US Government Accounting Office, for example, showed that the average US government agency spent more than US$220 per certificate."

[82] See http://en.wikipedia.org/wiki/Web_of_trust
[83] See http://www.voltage.com/

Validating the origin of public keys is not a problem unique to PGP; all crypto systems must address this issue. Ultimately, it will often come down to the question of whether or not you are comfortable with someone else (i.e. a CA) making your trust decisions for you, even if that person is in a better position and has more information available to them to do so. The key point to remember is that PGP can only be as secure as the way it is implemented[84].

When using PGP to encrypt data, the first thing it does is to compress the plaintext. Compression reduces the size of the encrypted file which in turn, places less load on network bandwidth and increases transmission speeds across the WAN, however these are not the only advantages. Compression increases the security of encrypted data because the most common techniques used to crack ciphertext exploit patterns found in the plaintext. Compression reduces these patterns, making it much harder to break[85].

PGP then creates a session key; a one time secret key that is randomly generated by the PGP software and used by the encryption algorithm to encrypt the plaintext; the result is ciphertext. Once the data is encrypted, the session key is encrypted using the recipient's public key (if symmetric encryption is being used, the agreed shared key is used). The public key encrypted session key is transmitted along with the ciphertext to the recipient. Decryption reverses this process. PGP running on the recipient's computer uses his or her private key (or the agreed shared key) to recover the one time session key, which PGP then uses to decrypt the ciphertext.

The combination of the two encryption methods (encrypting the data with a session key, known as conventional encryption, and then encrypting the session key with PKI) introduces both speed and security to the encryption process. As noted by J. Wilson in an article focusing on data privacy and published on the Adobe website, “Conventional encryption/decryption is extremely strong

[84] Some documented PGP ‘horror stories’ can be found at http://home.clara.net/heureka/sunrise/pgphorr.htm
[85] See http://www.infoanarchy.org/en/Cryptography

…..and about 1000 times faster than public-key encryption."[xix] This provides speed as PGP quickly encrypts the data. Public key encryption provides a secure means of key distribution.

PGP Usage to Secure Data

The above is a very high level overview of PGP and does not cover many of the security vulnerabilities that affected certain implementations of early versions of the technology[86], or any of the hash functions that PGP utilises to ensure integrity of the data. What it does do is highlight the fact that PGP is a credible option for file encryption and something that could be used as part of a solution to securely transfer files over the network.

"PGP's popularity and usage has grown to make it the de-facto standard for email encryption."[xx] As a result, PGP development and its support for new technologies continues to evolve. For example, PGP products are available to support email encryption on handheld devices like Blackberries[87]. Numerous books and online articles exist that provide in depth reading on PGP. Recommended reading is Simon Garfinkel's "PGP: Pretty Good Privacy" (http://www.oreilly.com/catalog/pgp/, ISBN 10: 1-56592-098-8), and http://www.pitt.edu/~poole/PGP.htm, and http://www.xs4all.nl/~josvg/pgs/pgp-how.html.

PGP is not a file transfer technology; it is, primarily, a file encryption technology. As a result, PGP could only form part of a solution to securely transfer data over the WAN; another protocol would be required to physically transfer data across the network. If that protocol were FTP, user credentials would still be sent in clear text.

When using PGP, remain in encrypted format when they are "at rest" on local hard drives,

---

[86] See http://news.zdnet.co.uk/communications/0,1000000085,2081034,00.htm and HTTPS://www.centr.org/docs/2000/10/iis-dnssecws-report.html,
[87] See http://www.pgp.com/products/pgp_support_package_for_bb/index.html

providing secure storage of data. PGP tools such as the PGP command line utility[88] will also support the automated encryption and digital signing of files through the execution of new commands in existing scripts.

PGP will only provide encryption; it does not support file transfer. Therefore, if PGP is used to encrypt data, the challenge of transferring that data over the network in a reliable and efficient manner remains. There are newer (although not necessarily better) technologies that provide similar data encryption features which should also be considered. One such technology that comes as standard with newer Windows based operating systems is Microsoft's Encryption File System (EFS).

EFS

Encryption File System (EFS) is a Microsoft proprietary technology that provides local file encryption and central key management and recovery. EFS's encryption technology, which can operate on a stand alone basis or as part of an Active Directory (AD) deployment, is integrated into the file system, meaning users are unable to access the hard disk without going through the file system (e.g. if a user were to boot a windows machine from a floppy disk to another OS, the EFS encrypted files would be unreadable). Encrypted files can only be decrypted and viewed by a user or application if they present the appropriate cryptographic key.[89]

EFS is provided free of charge by Microsoft but is limited by the fact that it is a component of the NTFS file system and subsequently only supported on the most recent Microsoft Operating System versions (Windows 2000, Windows XP Professional, Windows Server 2003 and Windows Vista). If a file that is protected by EFS is moved or copied to a non-EFS file system, it will be decrypted. Typically, USB thumb drives and other mobile devices like PDA's and MP3 players do not

[88] See http://www.pgp.com/products/commandline/
[89] A detailed overview of EFS is available at http://www.microsoft.com/technet/security/guidance/cryptographyetc/efs.mspx

use the NTFS file system (normally they use FAT). This means that users are able to easily decrypt their files and transport them in an insecure manner.

When EFS was first released by Microsoft, the network administrator had no ability to centrally control which file types or directories were encrypted. Although the actual encryption and decryption of data was transparent to the user (because EFS operates at the file system level), the onus for effective use of EFS was very much placed on the user. They were responsible for identifying which folders would be encrypted, and then ensuring that appropriate files were moved or saved to those folders. This issue is discussed by E. Jans, writing for Biztechmagazine.com in 2007, where he says "you could just direct users to encrypt classified information, or create an encrypted folder and require that all users place their classified data into that folder. However, these measures rely on the user's ability to realize what data is classified, and to follow through on the directive and handle it appropriately. Leaving security to user discretion is frequently a recipe for disaster."[xxi]

In May 2007 Microsoft released the Data Encryption Toolkit (DET) for Mobile PC's[90] which predominantly provided guidance and tools to help users protect data on mobile devices. The DET also included the EFS Assistant, a tool that allows network administrators to control (via AD and GPO) exactly which files and directories on user machines are to be encrypted. This new functionality, combined with the existing fact that users can be auto-registered to receive EFS certificates through AD (where a PKI and CA is in place), meant that the whole EFS encryption piece is now totally transparent to the user.

Similar to the way PGP works, EFS encrypts data with a symmetric key (conventional encryption) and then encrypts that key with the public key of a PKI pair. The key is bound to the users Windows profile and only accessible by that user after valid logon credential have been provided.

---

[90] See http://www.microsoft.com/downloads/details.aspx?FamilyID=1a99576a-fe67-418f-88b1-81e2055fe977&DisplayLang=en

In order to decrypt the data, the private key of the PKI pair must be available. "If the private key is damaged or missing, even the user that encrypted the file cannot decrypt it."[xxii]

File encryption keys can be exported to CD or USB key, or can be stored within a users Active Directory (AD) profile. AD also supports the creation of Key Recovery Agents (KRA) to enable the recovery of all keys in a given AD Operating Unit (OU) should keys become lost or damaged.

Because the user's public and private keys are stored in the local user profile on their laptop/PC, those keys are only as secure as the password that is used to protect that profile. A weak password will mean weak protection of the keys. AD GPO's that control password length (e.g. 8 characters or more) and complexity (e.g. must contain an alpha numeric or special character and a capital letter) in addition to user education are key to ensuring the safety of the profile and access to the machine.

Steps to deploy EFS differ based on the operating system you are using (Microsoft KB article KB222054[91] details the process for Windows 2000; KB article 307877[92] details the process for Windows XP), and certain functionality is only supported in later OS versions (e.g., per KB article KB223316[93], it is not possible to encrypt a file for multiple users in Windows 2000). This can make deploying EFS across a homogeneous operating system environment challenging.

When using EFS, as is the case with any file encryption technology, there is always the risk that irrespective of how careful a user or system administrator has been to create private key backups and assign KRA's, a set of circumstances may arise that means encrypted data on a users hard disk cannot be decrypted and is lost for good. As a user running EFS to encrypt multiple file types and directories on his laptop, the author of this paper experienced such circumstances in mid 2007 after suffering a

[91] See http://support.microsoft.com/default.aspx?scid=kb;en-us;222054&sd=tech
[92] See http://support.microsoft.com/default.aspx?scid=kb;en-us;307877&sd=tech
[93] See http://support.microsoft.com/kb/223316

hard drive failure. Using an offline backup of the EFS private key to successfully decrypt all data that was initially recovered from undamaged sectors of the disk, it became apparent that some data was missing. The only way to recover this data was to pull it from lost clusters on the drive. After doing this, files that were originally stored in clear text were readable however, as you would expect, all files that were encrypted using EFS were not. The problem arose after moving these encrypted files to a rebuilt laptop, which had a copy of the correct private EFS key that was originally used to encrypt the files, and trying to open them. When the recovered data was pulled from the hard disk clusters and moved to the new laptop it appeared that the encryption attribute was no longer set. As a result, EFS did not realize the files required decryption prior to opening, so they were opened in a completely unreadable format and unfortunately, lost forever. While this example is not meant to deter the reader from implementing data encryption across an enterprise, it is important to be aware that such problems can, and so, arise (and that all directories on a laptop should be regularly backed up!).

Microsoft has clearly stated that "EFS isn't designed to protect data while it's transferred from one system to another."[xxiii] Because EFS will automatically decrypt data when moving or copying files to a system that does not support EFS, such a solution could end up transmitting data in clear text (which PGP would never do). As a result, it may be more appropriate to view EFS as an extra layer in an overall defense in depth strategy (i.e. local encryption of files) as opposed to a specific tool to meet an organizations data transfer needs.

"Off the Shelf" data encryption tools

With the rapid growth in demand for data encryption technologies, it is not surprising that numerous vendors exist in the marketplace that provide off the shelf solutions for data encryption. Such vendors (e.g. Pointsec[94], SafeBoot[95]) have multiple products that essentially provide the same

[94] See http://www.checkpoint.com/pointsec/

functionality as PGP and EFS, that can be used to protect different data storage mediums. For example Pointsec, now part of the Checkpoint organization, has separate data security solutions that can be centrally deployed and managed to provide full disk encryption with access control (Pointsec PC[96]), and to encrypt data on removable media devices such as USB memory sticks (Pointsec Protector[97]) and mobile devices such as Pocket PC's and Smartphones (Pointsec Mobile[98]). SafeBoot has similar products.

In organizations where data encryption tools like PGP, EFS or Pointsec etc are used to encrypt data, it is important to note that it will never be possible to fully guarantee the security of that data. Password cracking tools like John the Ripper[99] can be used to crack encryption keys, especially when those keys are applied to weaker algorithms like DES. Such tools perform brute force attacks on encrypted data with the ability to run up to 200 keys a second from just a normal desktop machine. While stronger algorithms like 3DES are in theory crackable, assuming a secure non dictionary string has been used (e.g. dhfjKISYDHkshf83612hdkslp!!@d#) realistically it would take years to do, or at least enough time to mean that the data is all but useless by the time it is decrypted. Although only exploitable by the most advanced attacker it is also necessary to consider the fact that encrypted data is not always held in encrypted format on the local host, especially when it is being handled by memory. "Regardless of the encryption used, the data must exist unencrypted at some point, usually in memory, where host-based 'wiretaps' can sniff it before it's encrypted".[xxiv] Sniffing unencrypted data from memory is difficult to do but certainly possible.

---

[95] See http://www.safeboot.com/; (at the time of writing this paper, McAfee announced its intentions to buy SafeBoot. See http://www.eweek.com/article2/0,1895,2193106,00.asp)
[96] See http://www.checkpoint.com/products/datasecurity/pc/index.html
[97] See http://www.checkpoint.com/products/datasecurity/protector/index.html
[98] See http://www.checkpoint.com/products/datasecurity/mobile/index.html
[99] See http://www.openwall.com/john/

Hardware Based Encryption

The data encryption tools that have been discussed in this paper to date are all software based. Hardware based encryption has traditionally been associated with devices that provide network functionality, i.e. plug-in data encryption modules that are installed in servers, routers, firewalls and load balancers to support the load that data encryption and decryption place on the processor. Cisco's Virtual Private Network (VPN) Advanced Integration Module (AIM) is one such example, and can be inserted into a number of Cisco products which they claim "provides up to 40 percent better performance for IPsec VPN over the built-in IPsec encryption, and up to twice the performance for SSL Web VPN encryption[100]". Data encryption of laptops has traditionally been the task of software however, with the recent release of full disk encryption hardware like Seagate Technology's Momentus 5400 FDE.2[101] this could be about to change. "Seagate is betting companies will embrace hardware-based encryption because of cost, performance and easier management[xxv]".

Cost, performance and easier management are just three of a number of issues that should to be considered by organizations when comparing hardware and software based encryption technologies on end user machines[102]. This paper will not discuss hardware based encryption technologies for laptops in detail however, recently released products like the Seagate Momentus 5400 FDE.2 has meant they should at least be considered as a viable option by organizations looking to encrypt data stored on end user machines.

[100] http://www.cisco.com/en/US/products/ps5853/products_data_sheet0900aecd804ff58a.html
[101] See http://searchsecurity.techtarget.com/magazineFeature/0,296894,sid14_gci1262498,00.html
[102] An interesting comparison of hardware vs. software encryption can be found at http://www.oa.missouri.gov/itsd/cio/architecture/domains/security/CC-CryptoHardwarevsSoftwareEncryption041304ARCapp.pdf.

5.Recommendation

Every solution has its positives, but more importantly, its negatives. As a result it does not appear that one single technology, on its own, will be able to fully meet our needs. A hybrid solution is the best answer.

For large enterprises with unlimited resource and highly sensitive data, it is recommended that an internal PKI be deployed to provide a trusted key distribution system, and PGP used to locally encrypt all files (or EFS in a Windows only environment) before they are transferred. Properly and securely configured and managed web servers would then support FTP over SSL (FTPS) for integrity and authentication as well as the actual transfer of data. All files would remain in encrypted format once they reach their destination.

A solution for smaller business with less resource and less sensitive data could involve any combination of any of the other technologies. One recommendation is to use Microsoft's EFS technology to encrypt data on the local hard drive, and then FTP over SSH to transfer the file. In non-Windows environments (common in educational institutions), PGP would be used for local file encryption.

A final option for both the larger and smaller enterprise could be to consider purchasing a 3rd party "off the shelf" product that inherently includes most of the technologies discussed in this paper. Here, functionality, compatibility and cost would be the main issues to contend with as the underlying technologies would be transparent to the user. While the initial financial outlay of this option may be higher than a self managed solution, ease of implementation and lower support overheads going forward, may make it more attractive and cost effective in the long run.

6. Summary

Transferring data across a WAN in unencrypted clear text format using protocols like FTP is an insecure and unacceptable solution. The challenge of securing data when it is "at rest", i.e. being stored on local server or client hard drives, must also be addressed.

The list of alternate solutions discussed in this paper (SSL, SSH, WebDAV, HTTPS, EFS, IPsec) each carry their own strengths and weaknesses.

FTP on its own is unacceptable due to the inherent security weaknesses that affect it (primarily the fact that all data is send in clear text) however, the protocol itself may be suitable to provide underlying file transfer functionality. FTP is reliable, easy to use, supports manual and automated transfers and has a free client that is included in almost all standard Operating Systems. In instances where FTP is used to transfer data, it would be necessary to identify suitable and compatible technologies to secure the data before, during and after transmission.

IPsec is not only used as a WAN technology to provide secure VPN tunnels between remote office locations, it is also used to encrypt data sent between individual clients on the internal network. The inbuilt flexibility of IPsec provides encryption, authentication and data integrity through the use of any combination of the encryptions protocols it supports. Although this flexibility means it can be challenging to configure and support, large organizations should have the necessary resources to implement such a solution. IPsec would also only encrypt data during transmission; files would remain stored on client and server hard drives in clear text.

FTP over SSH (SFTP) also creates issues. SFTP does not address the secure storage of files on network systems when they are not being transmitted, and it is also very easy to deploy insecurely. In

addition to this, many SSL programs will by default, only tunnel the initial FTP control session, meaning that the main file transfer which is done over a separate data session will remain unencrypted plain text FTP.

FTP over SSL/TLS (FTPS) is another possible solution. Because TLS runs at a lower level of the 7 Layer OSI model to FTP, all traffic is securely encapsulated without the need to have a client route traffic through separate connections as seen with FTP over SSH. FTP over SSL is reasonably easy to implement as most FTP Servers, and more recently clients, support it. However, the SSL certificates can be cumbersome and expensive to purchase and install. When using FTPS, files would also remain stored on client and server hard drives in clear text.

WebDAV over SSL could be used but WebDAV is subject to a number of security vulnerabilities. As a result, WebDAV is another technology that should only be used for transporting data over the network; a separate technology would be required to secure it. In such instances EFS could be used, however, because EFS is a Microsoft proprietary protocol, this solution could only be implemented in a Windows only environment. EFS would also automatically decrypt data sent to a non-NTFS formatted file system. To overcome these problems, PGP could be used to encrypt the data instead of EFS.

PGP provides both confidentiality and integrity (non-repudiation) of data, but presents the challenge of key management. The responsibility of securing and trusting keys can be given users to create a "web of trust", which has obvious risks, or a PKI needs to be implemented which, while a possibility for large corporations with sufficient budget and in-house skills, is not a viable option for the smaller and medium sized business. If FTP were to be used as the underlying technology to transfer the (now encrypted) file, the original username and password for the transfer would still be sent in clear text.

Any decision regarding which technology is to be implemented will depend heavily on the

existing IT environment in place throughout the organization (e.g. if 50% of users have Mac's it would not be possible to implement EFS), available budget, the knowledge and experience that network administrators who will manage the solution have of the protocols discussed in this paper, and of course the amount of use and number of users that will use the solution on a daily basis.

7. References

[i] CIA triad. Wikipedia. Retrieved August 14, 2007, from http://en.wikipedia.org/wiki/CIA_triad

[ii] Bigelow, S.J. (2005). On Eagle's Wings. Processor.com, General Information, Vol.27 Issue 16. Retrieved August 14, 2007, from http://www.processor.com/editorial/article.asp?article=articles%2Fp2716%2F09p16.asp

[iii] Evans, T. D. (2002). NetBios, NetBEUI, NBF, SMB, CIFS Networking. Retrieved September 22, 2007, from http://ourworld.compuserve.com/homepages/TimothyDEvans/intro.htm#History

[iv] Performance of Microsoft's File Sharing over Wide Area Networks. Version 1.0. Retrieved October 5, 2007 from http://www.scps.org/Documents/SMB_OVER_WAN_1.0.doc

[v] Hughes, T. (2003). File sharing over the WAN - Storage Networking. *Computer Technology Review*. Retrieved September 22, 2007, from http://findarticles.com/p/articles/mi_m0BRZ/is_3_23/ai_99751341

[vi] ISA Server 2000 Feature Pack 1. Microsoft TechNet. Version 1. Retrieved September 22, 2007, from http://www.microsoft.com/technet/isa/2000/isafp1/epo.mspx

[vii] Manion, A. (2002, last updated July 3, 2003). Multiple vendors' firewalls do not adequately keep state of FTP traffic. US-CERT. Vulnerability Note VU#328867. Document revision 74. Retrieved September 22, 2007, from http://www.kb.cert.org/vuls/id/328867

[viii] Golen, P. (2003, last updated July 22, 2004). SSH. Retrieved May 12, 2007, from http://www.windowsecurity.com/articles/SSH.html

[ix] Zeigler, R. L. (2002). Linux Firewalls. Second Edition. New Riders.

[x] Secure File Transfers. SSH Communications Security. Retrieved August 15, 2007, from http://www.ssh.com/solutions/applications/sftp.html#answer7

[xi] About WebDav (updated 2005). Microsoft TechNet. Retrieved September 22, 2007, from http://technet2.microsoft.com/windowsserver/en/library/9a772ae5-3864-411e-98d6-9700eb3acb8c1033.mspx?mfr=true

[xii] 2006 University Information Technology Services student survey. (2006). Indiana University. Retrieved

September 26, 2007, from http://www.indiana.edu/~uitssur/2006/iub/iub_text06.html

[xiii] Connecting to the School of Social Work Network Server using Internet Explorer. University of Southern California, Information Technology Support. Retrieved September 26, 2007 from http://sowkweb.usc.edu/it/toWebDAV.php

[xiv] WebDAV Mini-Redirector (MRXDAV.SYS) Versions and Issues List. (2007). Greenbytes.de. Retrieved May 30, 2007, from http://www.greenbytes.de/tech/webdav/webdav-redirector-list.html

[xv] What is IPsec? (2003). Microsoft TechNet. Retrieved June 16, 2007, from http://technet2.microsoft.com/windowsserver/en/library/2a2f7792-5a4a-438b-8711-23694ae56e3a1033.mspx?mfr=true

[xvi] Nahra, K. J. (2007). White House ID Theft Task Force Releases Strategic Plan: What Do Businesses Need to Know? *Privacy in Focus*. June 2007.

[xvii] Rogers, M. (2005). The Insider Threat: Debunking the 'Wagon Wheel' approach to information security. SearchSecurity.com. Retrieved January 14, 2007, from http://searchsecurity.techtarget.com/columnItem/0,294698,sid14_gci1064080,00.html

[xviii] Martin, L. (2006). Fitting Square Pegs into Round Holes. *IEEE Security & Privacy*, vol. 4, no. 5, September/October 2006, pp. 64-66.

[xix] Wilson, J. Data Privacy - Security In An Insecure World. Adobe.com. Retrieved September 23, 2007, from http://www.adobe.com/devnet/server_archive/articles/data_privacy.html

[xx] Atkins, D. PGP Development Team, MIT. Taken from media review of Garfinkel, S. (1994). PGP: Pretty Good Privacy. O'Reilly. Retrieved September 17, 2007, from http://www.oreilly.com/catalog/pgp/

[xxi] Jans, E. (2007). Encrypting Data With Vista. Biztechmagazine.com. Retrieved September 23, 2007, from http://www.biztechmagazine.com/print_friendly.asp?item_id=225

[xxii] Bragg, R. The Encrypting File System. Microsoft TechNet. Retrieved September 23, 2007, from http://www.microsoft.com/technet/security/guidance/cryptographyetc/efs.mspx

[xxiii] Bragg, R. The Encrypting File System. Microsoft TechNet. Retrieved September 23, 2007, from

http://www.microsoft.com/technet/security/guidance/cryptographyetc/efs.mspx

[xxiv] Hoglund, G. (2007). Pwned. *Information Security*. September 2007

[xxv] Roiter, N. (2007). Hard Core. *Information Security*. July 2007.