

Beyond FTP...



*Securing and Managing
Data Transmissions*



Company Background

- Founded in 1994
- Based in Nebraska
- Private company with no Outside Funding
- Dedicated to Research and Development
- IBM Advanced Business Partner
- Member of IBM Developer's Program
- Microsoft Business Partner
- Responsive Toll-Free Technical Support (including off-hours Emergency Support)

Linoma Software provides innovative technologies for automating and securing data movement. Linoma Software has a diverse install base of customers around the world including Fortune 500 companies, non-profit organizations and government entities.

Intro and Agenda

Panelists:

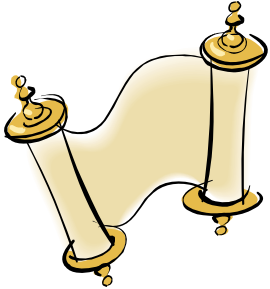
- Bob Luebbe – Chief Architect for Linoma Software
- Sai Pullabhotla – Senior Architect for Linoma Software (FTP guru)

Agenda:

- Vulnerabilities with traditional FTP
- OpenPGP Encryption
- Introduction to Secure FTP (SFTP and FTPS)
- Authentication - Key and certificate management
- Advantages/disadvantages between SFTP and FTPS
- Error notification and logging
- Introduction to GoAnywhere product for automating and securing FTP

* Feel free to enter questions in control panel

The 3 Truths about Standard FTP



1. You cannot always be sure that the entity with whom you are communicating is really who you think it is.
2. FTP data can be intercepted, so it is possible that it can be read by an unauthorized third party (attacker)
3. If an attacker can intercept the data, they may be able to modify the data before sending it on to the receiver.

How can your transmissions be monitored?

- Network sniffer tools
- Non-switched "shared" hubs (which pass traffic around the network)
- Unsecured wireless networks
- PC remote access software (e.g. gotomypc) to gain access to internal network
- IP spoofing (hacker pretends to be a "trusted" partner)
- Router vulnerabilities
- Internet Service Providers (ISPs)



Sniffing FTP using Wireshark (capturing user/password)

The screenshot shows a Wireshark capture of an FTP session. The packet list pane displays the following key packets:

No.	Time	Source	Destination	Protocol	Info
8	0.656512	216.170.	192.168.	FTP	Request: USER jdoe
9	0.657782	192.168.	216.170.	FTP	Response: 331 Enter password.
10	0.768250	216.170.	192.168.	FTP	Request: PASS jdoe1122
11	0.768547	192.168.	216.170.	FTP	Response: 230 JDOE logged on.

Red arrows in the original image point to the 'USER jdoe' and 'PASS jdoe1122' fields in the packet details pane, with the text 'User and password in the clear' overlaid in red.

Microsoft: <live capture in progress> File: ... Packets: 48 Displayed: 48 Marked: 0 Profile: Default

Sniffing FTP using Wireshark (capturing data)

Microsoft: Capturing - Wireshark

File Edit View Go Capture Analyze Statistics Help





Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
10	1.109504	216.170.192.168	192.168.216.170	FTP	Response: 213 OS/400 is the remote operating system. The TCP/IP version is VJ0400.
17	1.109989	192.168.216.170	216.170.192.168	FTP	Request: SITE LISTFMT 1
18	1.214281	216.170.192.168	192.168.216.170	FTP	Response: 250 Directory listing format (LISTFMT) option set to 1.
19	1.214567	192.168.216.170	216.170.192.168	FTP	Request: SITE NAMEFMT 1
20	1.320570	216.170.192.168	192.168.216.170	FTP	Response: 250 Now using naming format "1".
21	1.321549	192.168.216.170	216.170.192.168	FTP	Request: Pwd
22	1.428405	216.170.192.168	192.168.216.170	FTP	Response: 257 "/QSYS.LIB/QGPL.LIB" is current library.
23	1.428861	192.168.216.170	216.170.192.168	FTP	Request: MKD /elichtas
24	1.531385	216.170.192.168	192.168.216.170	FTP	Response: 550 Directory /elichtas already exists. Directory not created.
25	1.531770	192.168.216.170	216.170.192.168	FTP	Request: TYPE A
26	1.637300	216.170.192.168	192.168.216.170	FTP	Response: 200 Representation type is ASCII nonprint.
27	1.638564	192.168.216.170	216.170.192.168	FTP	Request: PORT 192,168,1,102,199,200
28	1.748407	216.170.192.168	192.168.216.170	FTP	Response: 200 PORT subcommand request successful.
29	1.748692	192.168.216.170	216.170.192.168	FTP	Request: STOR /elichtas/DATA.txt
30	1.880565	216.170.192.168	192.168.216.170	TCP	ftp-data > 51144 [SYN] Seq=0 win=32768 Len=0 MSS=1412 WS=0 TSV=249987000 TSER=0
31	1.880685	192.168.216.170	216.170.192.168	TCP	51144 > ftp-data [SYN, ACK] Seq=0 Ack=1 win=8192 Len=0 MSS=1460 WS=8 TSV=991325 TSER=249987000
32	1.992402	216.170.192.168	192.168.216.170	TCP	ftp-data > 51144 [ACK] Seq=1 Ack=1 win=32768 Len=0 TSV=249987500 TSER=991325
33	2.033475	216.170.192.168	192.168.216.170	TCP	ftp > 51143 [PSH, ACK] Seq=606 Ack=160 win=8192 Len=0
34	2.054401	216.170.192.168	192.168.216.170	FTP	Response: 150 Sending file to /elichtas/DATA.txt
35	2.055397	192.168.216.170	216.170.192.168	FTP-DATA	FTP Data: 337 bytes
36	2.055675	192.168.216.170	216.170.192.168	TCP	51144 > ftp-data [FIN, ACK] Seq=338 Ack=1 win=16640 Len=0 TSV=991342 TSER=249987500
37	2.179344	216.170.192.168	192.168.216.170	TCP	ftp-data > 51144 [ACK] Seq=1 Ack=339 win=32431 Len=0 TSV=249987500 TSER=991342
38	2.180273	216.170.192.168	192.168.216.170	TCP	ftp-data > 51144 [FIN, PSH, ACK] Seq=1 Ack=339 win=32768 Len=0 TSV=249987500 TSER=991342
39	2.180311	192.168.216.170	216.170.192.168	TCP	51144 > ftp-data [ACK] Seq=339 Ack=2 win=16640 Len=0 TSV=991355 TSER=249987500
40	2.249259	192.168.216.170	216.170.192.168	TCP	51143 > ftp [ACK] Seq=160 Ack=646 win=16128 Len=0
41	2.366464	216.170.192.168	192.168.216.170	FTP	Response: 226 File transfer completed successfully.
42	2.367544	192.168.216.170	216.170.192.168	FTP	Request: QUIT
43	2.501435	216.170.192.168	192.168.216.170	TCP	ftp > 51143 [PSH, ACK] Seq=689 Ack=166 win=8192 Len=0
44	2.503360	216.170.192.168	192.168.216.170	FTP	Response: 221 QUIT subcommand received.

0000 00 18 f8 42 35 eb 00 19 d2 86 ce dd 08 00 45 00 ...B5... ..E.
0010 01 85 0a 4c 40 00 80 06 45 83 c0 a8 01 66 d8 aa ...L@... E...f..
0020 0e eb c7 c8 00 14 4a 46 48 78 ee 26 89 60 80 18JF Hx.&...
0030 00 41 5b b6 00 00 01 01 08 0a 00 0f 20 6e 0e e6 .A[..... ..n..
0040 81 ac 4a 6f 68 6e 20 44 6f 65 2c 33 33 33 2d 32 ..John D oe,333-2
0050 32 2d 34 34 34 34 2c 43 43 4e 34 34 34 34 35 35 2-4444,C CN444455
0060 35 35 36 36 36 36 37 37 37 37 2c 31 31 2f 32 30 55666677 77,11/20
0070 2f 31 39 38 33 0d 0a 4a 61 6e 65 20 44 6f 65 2c /1983..J ane Doe,
0080 38 38 38 2d 38 38 2d 38 38 38 38 2c 43 43 4e 31 888-88-8 888,CCN1
0090 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 2c 23456789 0123456,
00a0 30 31 2f 30 31 2f 31 39 35 30 0d 0a 4a 61 63 6b 01/01/19 50..Jack
00b0 20 4a 61 63 6b 73 6f 6e 2c 31 31 31 2d 31 31 2d Jackson ,111-11-
00c0 31 31 31 31 2c 43 43 4e 36 35 34 33 32 31 30 39 1111,CCN 65432109
00d0 38 37 36 35 34 33 32 31 2c 30 38 2f 31 32 2f 31 87654321 ,08/12/1
00e0 39 37 37 0d 0a 52 69 63 6b 20 52 69 63 68 61 72 977..Ric k Richar
00f0 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99


Microsoft: <live capture in progress> File: ... Packets: 48 Displayed: 48 Marked: 0 Profile: Default

Data Which Needs a High Level of Protection

- Anything that is confidential to the organization, its employees and its customers
- Credit card numbers    
- Personal Identity Information (e.g. Social Security Numbers, Birthdates, etc.)
- Payroll information (e.g. wages)
- Health-related information
- Bank Account numbers
- Driver License numbers
- Financial data
- Trade Secrets (e.g. product formulas)



Why Should You Protect This Data?

- To comply with regulations:
 - HIPAA
 - Sarbanes Oxley
 - Gramm-Leach-Bliley Act
 - State privacy laws
- To avoid potential penalties and lawsuits
- To comply with PCI Security Standards 
- To avoid bad public relations
- To ensure your continued employment (you don't want to be the one that "takes the fall")

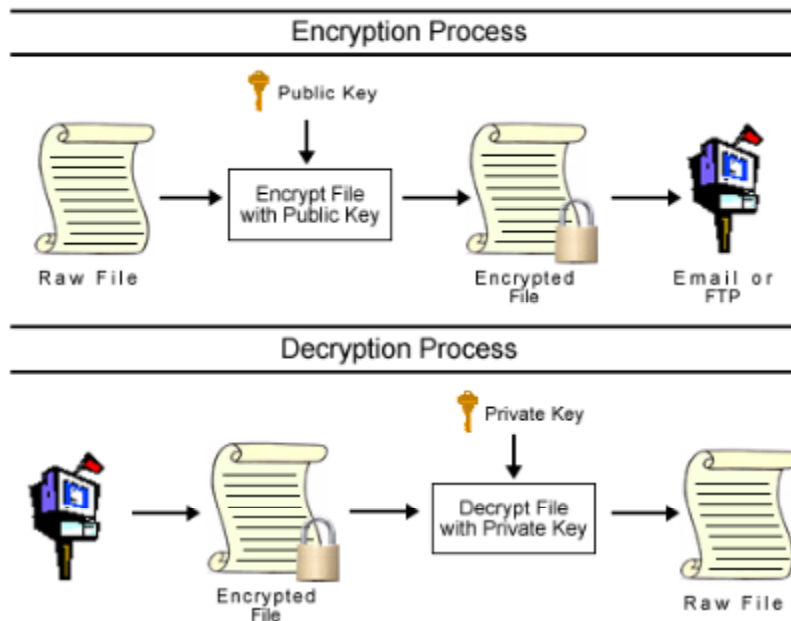
"A senior database administrator at a subsidiary of Fidelity National Information Services took data belonging to as many as 8.5 million consumers. The stolen data included names, addresses, birth dates, bank account and credit card information, the company said."

(Source: ComputerWorld, July 2007)



OpenPGP File Encryption

- Widely used for protecting files to be sent over the internet.
- Uses combination of Asymmetric-key and Symmetric-key cryptology to provide high level of protection and speed
- Encrypt with Public Key -- Decrypt with Private Key (Secret Key)



Caution: If an encrypted file is sent over a standard FTP connection, the FTP user ids, passwords and commands are still in the clear.

TERMS

OpenPGP standard is a non-proprietary and industry-accepted protocol which defines the standard format for encrypted messages, signatures and keys.

Private Key is the portion of a Key Pair which is used by the owner to decrypt information and to encode digital signatures. The Private key, typically protected by a password, should be kept secret by the owner and NOT shared with trading partners. Also known as a Secret Key.

Public Key is the portion of the Key Pair which is used to encrypt information bound for its owner and to verify signatures made by its owner. The owner's Public key should be shared with its trading partners.

The 3 Main Benefits of Secure FTP

1. Secure FTP allows computer systems to ensure the identity of each other (Authentication).
2. Secure FTP creates an encrypted connection between computer systems, preventing the interception of users, passwords and data (Data Privacy).
3. Secure FTP implements hash functions to ensure that data was not modified in transit (Data Integrity).

Two Types of Secure FTP

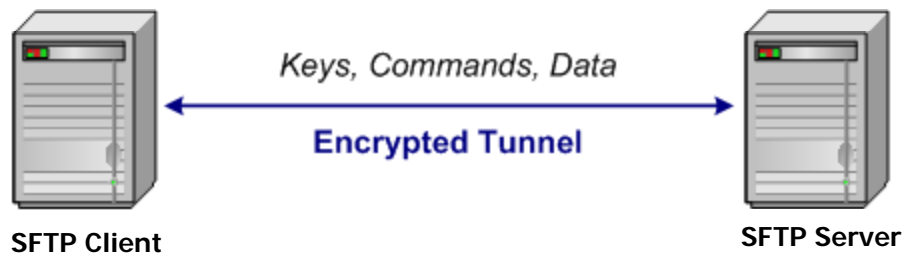
- **SFTP** = FTP over SSH
- **FTPS** = FTP over SSL/TLS

SFTP

- SFTP = FTP over SSH
- SSH 2.0 is latest standard
- Sometimes referred to as SCP 2.0
- SFTP encrypts the entire connection
- Authenticate using a user id and either a
 - Password and/or
 - Public key
- Popular in UNIX and LINUX systems
- Trading partner must have a SFTP server (in order to connect to them)
- Most FTP commands are supported

TERMS

SSH is an abbreviation for Secure Shell. SSH is both a computer program and an associated network protocol designed for encrypting communications between two untrusted hosts over a network. It utilizes Public keys to provide asymmetric cryptology.



Sniffing SFTP using Wireshark (key exchange)

The image shows a Wireshark network traffic capture of an SSH session. The packet list pane displays the following entries:

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.216.170	216.170.192.168	TCP	51180 > ssh [SYN] Seq=0 win=8192 Len=0 MSS=1460 WS=8
2	0.176820	216.170.192.168	192.168.216.170	TCP	ssh > 51180 [SYN, ACK] Seq=0 Ack=1 win=8192 Len=0 MSS=1412 WS=0
3	0.176953	192.168.216.170	216.170.192.168	TCP	51180 > ssh [ACK] Seq=1 Ack=1 win=16896 Len=0
4	0.239797	192.168.216.170	216.170.192.168	SSH	Client Protocol: SSH-2.0-1.0\r
5	0.481663	216.170.192.168	192.168.216.170	SSHv2	Server Protocol: SSH-1.99-openssh_3.5p1
6	0.482468	192.168.216.170	216.170.192.168	SSHv2	Client: Key Exchange Init
7	0.646896	216.170.192.168	216.170.192.168	SSHv2	Server: Key Exchange Init ← Key Exchange
8	0.752882	192.168.216.170	216.170.192.168	SSHv2	Client: Diffie-Hellman Key Exchange Init
9	0.934805	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=568 Ack=1046 win=8192 Len=0
10	0.951949	216.170.192.168	216.170.192.168	SSHv2	Server: New Keys
11	0.999645	192.168.216.170	216.170.192.168	SSHv2	Client: New Keys
12	1.353535	216.170.192.168	192.168.216.170	TCP	ssh > 51180 [PSH, ACK] Seq=1032 Ack=1062 win=8192 Len=0
13	1.353597	192.168.216.170	216.170.192.168	SSHv2	51180 > ssh [PSH, ACK] Seq=1062 Ack=1032 win=15872 Len=48[Malformed Packet]
14	1.483677	216.170.192.168	192.168.216.170	TCP	ssh > 51180 [PSH, ACK] Seq=1032 Ack=1110 win=8192 Len=0
15	1.485488	216.170.192.168	216.170.192.168	SSHv2	ssh > 51180 [PSH, ACK] Seq=1032 Ack=1110 win=8192 Len=48[Malformed Packet]
16	1.486328	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=80
17	1.657632	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1080 Ack=1190 win=8192 Len=0
18	1.658693	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=32
19	1.673250	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=56
20	1.855710	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1112 Ack=1246 win=8192 Len=0
21	1.858707	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=48
22	1.892786	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=56
23	2.028834	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1160 Ack=1302 win=8192 Len=0
24	2.285679	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=40
25	2.286416	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=48
26	2.406631	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1200 Ack=1350 win=8192 Len=0
27	2.409480	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=48
28	2.410250	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=48
29	2.558708	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1248 Ack=1398 win=8192 Len=0
30	2.564611	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=88
31	2.566307	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=56
32	2.758703	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1336 Ack=1454 win=8192 Len=0
33	2.765585	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=80
34	2.767048	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=80
35	2.878776	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1416 Ack=1534 win=8192 Len=0
36	2.885502	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=56
37	2.888067	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=400
38	3.055630	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1472 Ack=1934 win=8192 Len=0
39	3.057557	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=64
40	3.065100	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=56
41	3.231889	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1536 Ack=1990 win=8192 Len=0
42	3.241594	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=64
43	3.243312	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=32
44	3.434880	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1600 Ack=2022 win=8192 Len=0

The status bar at the bottom indicates: Microsoft: <live capture in progress> File: ... Packets: 52 Displayed: 52 Marked: 0 Profile: Default

Sniffing SFTP using Wireshark (capturing data)

The screenshot shows the Wireshark interface with a capture of an SSH session. The packet list pane contains the following data:

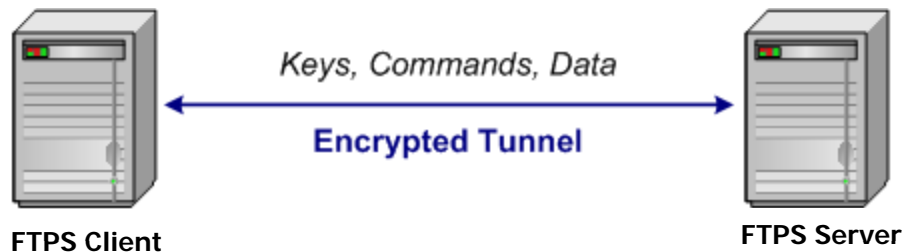
No.	Time	Source	Destination	Protocol	Info
10	0.951949	216.170.192.168	192.168.216.170	SSHv2	Server: New Keys
11	0.999645	192.168.216.170	216.170.192.168	SSHv2	Client: New Keys
12	1.353535	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1032 Ack=1062 win=8192 Len=0
13	1.353597	192.168.216.170	216.170.192.168	SSHv2	51180 > ssh [PSH, ACK] Seq=1062 Ack=1032 win=15872 Len=48[Malformed Packet]
14	1.483677	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1032 Ack=1110 win=8192 Len=0
15	1.485488	216.170.192.168	216.170.192.168	SSHv2	ssh > 51180 [PSH, ACK] Seq=1032 Ack=1110 win=8192 Len=48[Malformed Packet]
16	1.486328	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=80
17	1.657632	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1080 Ack=1190 win=8192 Len=0
18	1.658693	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=32
19	1.673250	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=56
20	1.855710	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1112 Ack=1246 win=8192 Len=0
21	1.858707	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=48
22	1.892786	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=56
23	2.028834	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1160 Ack=1302 win=8192 Len=0
24	2.285679	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=40
25	2.286416	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=48
26	2.406631	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1200 Ack=1350 win=8192 Len=0
27	2.409480	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=48
28	2.410250	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=48
29	2.558708	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1248 Ack=1398 win=8192 Len=0
30	2.564611	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=88
31	2.566307	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=56
32	2.758703	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1336 Ack=1454 win=8192 Len=0
33	2.765585	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=80
34	2.767048	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=80
35	2.878776	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1416 Ack=1534 win=8192 Len=0
36	2.885502	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=56
37	2.888067	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=400
38	3.055630	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1472 Ack=1934 win=8192 Len=0
39	3.057557	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=64
40	3.065100	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=56
41	3.231889	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1536 Ack=1990 win=8192 Len=0
42	3.241594	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=64
43	3.243312	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=32
44	3.424899	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1600 Ack=2022 win=8192 Len=0
45	3.424964	192.168.216.170	216.170.192.168	SSHv2	Encrypted request packet len=32
46	3.550765	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=56
47	3.711776	216.170.192.168	216.170.192.168	SSHv2	Encrypted response packet len=64
48	3.711868	192.168.216.170	216.170.192.168	TCP	51180 > ssh [ACK] Seq=2054 Ack=1720 win=16640 Len=0
49	3.712621	192.168.216.170	216.170.192.168	TCP	51180 > ssh [FIN, ACK] Seq=2054 Ack=1720 win=16640 Len=0
50	4.011016	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [PSH, ACK] Seq=1720 Ack=2055 win=8192 Len=0
51	4.013086	216.170.192.168	216.170.192.168	TCP	ssh > 51180 [FIN, PSH, ACK] Seq=1720 Ack=2055 win=8192 Len=0
52	4.013120	192.168.216.170	216.170.192.168	TCP	51180 > ssh [ACK] Seq=2055 Ack=1721 win=16640 Len=0

Encrypted Data

Microsoft: <live capture in progress> File: ... Packets: 52 Displayed: 52 Marked: 0 Profile: Default

FTPS

- FTPS = FTP over SSL/TLS
- Protects entire connection, including data, users passwords, commands, etc.
- Authenticate using X.509 certificates
- Explicit FTPS – Normal FTP available, but client can explicitly request server to switch to SSL/TLS
- Implicit FTPS – Server forces client to use SSL/TLS from the initial connection (ask your trading partner if they support this first)
- Trading partner must have an FTP server enabled for SSL/TLS



TERMS

Authentication is a mechanism to positively identify the server, and optionally the client, by requesting credentials such as a password or a digital signature.

Certificate is a digital identification document that allow both servers and clients to authenticate each other. A certificate contains information about a company and the organization that signed the certificate (such as Verisign).

SSL is an abbreviation for Secure Sockets Layer. SSL is a security protocol for encrypting communications between two hosts over a network. SSL utilizes certificates to establish trust between the two hosts.

TLS is the abbreviation for Transport Layer Security and is the successor to SSL.

Sniffing FTPS using Wireshark (capturing data)

Microsoft: Capturing - Wireshark

File Edit View Go Capture Analyze Statistics Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
44	2.812735	192.168.	216.170.	FTP	Request: \027\003\001\000\025\033\313\362\030-\336\221\253\260\312Py\320\337x\215+\332\204ue
45	3.009068	216.170.	192.168.	TCP	ftp > 51118 [PSH, ACK] Seq=1974 Ack=611 win=8192 Len=0
46	3.015058	216.170.	192.168.	FTP	Response: \027\003\001\000>\350\337\2264\361\257A\256q\006c\223(\235\253\273n/\340)\202\68\354\035\336\307\00!
47	3.015818	192.168.	216.170.	FTP	Request: \027\003\001\000\037\251\277\263\2331\277v\206\324\363\240\361\333\234f,\036\267\026c\261v\370\357\00!
48	3.128221	216.170.	192.168.	TCP	ftp > 51118 [PSH, ACK] Seq=2041 Ack=647 win=8192 Len=0
49	3.131083	216.170.	192.168.	FTP	Response: \027\003\001\000R8\341\212s\327s\206c\315\375\356\0245\221\312\364\221\313\233\273\201fd9\3121\350NI
50	3.131611	192.168.	216.170.	FTP	Request: \027\003\001\000\030
51	3.240240	216.170.	192.168.	TCP	ftp > 51118 [PSH, ACK] Seq=2128 Ack=676 win=8192 Len=0
52	3.244500	216.170.	192.168.	FTP	Response: \027\003\001\000<\313~\272\360\207\001gdreI>\344b\ +\262\275c\370\271 \367sL\034\365\205\222a\237\31:
53	3.245486	192.168.	216.170.	FTP	Request: \027\003\001\000\026\273G\020\241\373\236p\230
54	3.350273	216.170.	192.168.	TCP	ftp > 51118 [PSH, ACK] Seq=2193 Ack=703 win=8192 Len=0
55	3.354019	216.170.	192.168.	FTP	Response: \027\003\001\000\213\017\233\367\223c@\326\227<6@w\306\343st\275^\027#q\2674w\323\247R\236#\272D\2!
56	3.354477	192.168.	216.170.	FTP	Request: \027\003\001\000)+\323\240C.\600\222gAN5\271\300\200\234\225\200"\346\024v\341\326\221\274\341\217e'o'
57	3.355809	192.168.	216.170.	TCP	51119 > 42335 [SYN] Seq=0 win=8192 Len=0 MSS=1460 WS=8
58	3.462228	216.170.	192.168.	TCP	ftp > 51118 [PSH, ACK] Seq=2266 Ack=749 win=8192 Len=0
59	3.464145	216.170.	192.168.	TCP	42335 > 51119 [SYN, ACK] Seq=0 Ack=1 win=32768 Len=0 MSS=1412 WS=0
60	3.464243	192.168.	216.170.	TCP	51119 > 42335 [ACK] Seq=1 Ack=1 win=16896 Len=0
61	3.505463	192.168.	216.170.	TCP	51119 > 42335 [PSH, ACK] Seq=1 Ack=1 win=16896 Len=109
62	3.568158	216.170.	192.168.	FTP	Response: \027\003\001\0008f'k:\352\367\$'\0\216\342\345\b\260\ a?]\207(\026\244w~\244x;\021\037\336\272\345\320\:
63	3.660326	216.170.	192.168.	TCP	42335 > 51119 [PSH, ACK] Seq=1 Ack=110 win=32659 Len=1240
64	3.662015	192.168.	216.170.	TCP	51119 > 42335 [PSH, ACK] Seq=110 Ack=1241 win=15616 Len=139
65	3.762453	192.168.	216.170.	TCP	51118 > ftp [ACK] Seq=749 Ack=2327 win=15872 Len=0
66	3.811855	216.170.	192.168.	TCP	42335 > 51119 [PSH, ACK] Seq=1241 Ack=249 win=32768 Len=0
67	3.811926	192.168.	216.170.	TCP	51119 > 42335 [PSH, ACK] Seq=249 Ack=1241 win=15616 Len=43
68	3.919038	216.170.	192.168.	TCP	42335 > 51119 [PSH, ACK] Seq=1241 Ack=292 win=32725 Len=43
69	3.920013	216.170.	192.168.	TCP	[TCP window update] 42335 > 51119 [PSH, ACK] Seq=1284 Ack=292 win=32768 Len=0
70	3.920231	192.168.	216.170.	TCP	51119 > 42335 [PSH, ACK] Seq=292 Ack=1284 win=15616 Len=358
71	3.920591	192.168.	216.170.	TCP	51119 > 42335 [FIN, PSH, ACK] Seq=650 Ack=1284 win=15616 Len=23
72	4.051075	216.170.	192.168.	TCP	[TCP Dup ACK 69#1] 42335 > 51119 [PSH, ACK] Seq=1284 Ack=292 win=32768 Len=0
73	4.059040	216.170.	192.168.	TCP	42335 > 51119 [PSH, ACK] Seq=1284 Ack=674 win=32387 Len=0
74	4.060166	216.170.	192.168.	TCP	42335 > 51119 [PSH, ACK] Seq=1284 Ack=674 win=32768 Len=23
75	4.060200	192.168.	216.170.	TCP	51119 > 42335 [RST, ACK] Seq=674 Ack=1307 win=0 Len=0
76	4.062144	216.170.	192.168.	TCP	42335 > 51119 [FIN, PSH, ACK] Seq=1307 Ack=674 win=32768 Len=0
77	4.064120	216.170.	192.168.	FTP	Response: \027\003\001\000;\321\245\305=@y\3559\217\252\276\005\025\300\017\214\350y\305\224\316#\177N\362\202'
78	4.065168	192.168.	216.170.	FTP	Request: \027\003\001\000\026\357xc\316\343\257=\364\231)\307\375 \215\305zH\3433\210\366
79	4.174352	216.170.	192.168.	TCP	ftp > 51118 [PSH, ACK] Seq=2391 Ack=776 win=8192 Len=0
80	4.175307	216.170.	192.168.	FTP	Response: \027\003\001\000\031N\205\276\032\376;\262\306q\345v\ a\000m\341\326\224\322N\thy
81	4.175744	192.168.	216.170.	FTP	Request: \027\003\001\000\026\034n\312\2372\241BN#\247\347\342\ a\2244:\235yyk\331D
82	4.279368	216.170.	192.168.	TCP	ftp > 51118 [PSH, ACK] Seq=2421 Ack=803 win=8192 Len=0
83	4.281362	216.170.	192.168.	FTP	Response: \027\003\001\000\215\262\220\345\246r\350\006\372qs.\202\214\264\266\006MNs\217\267N\207\265\300\34:
84	4.282370	192.168.	216.170.	FTP	Request: \025\003\001\000\022\035\205\204\000\025=\374;\270e\314\325\260\205qH/O
85	4.284362	216.170.	192.168.	FTP	Response: \025\003\001\000\022+\030\323\327\352\200%\0161\216,\2010\035<\374C
86	4.284481	192.168.	216.170.	TCP	51118 > ftp [ACK] Seq=826 Ack=2497 win=15616 Len=0
87	4.284646	192.168.	216.170.	TCP	51119 > ftp [ACK] Seq=826 Ack=2497 win=15616 Len=0

Microsoft: <live capture in progress> File: ... Packets: 90 Displayed: 90 Marked: 0 Profile: Default

Creating your Certificate

1. Use a SSL certificate manager
2. Create a Certificate in your key store.
3. If your trading partner requires that your certificate is signed by a third party (not self-signed), then follow the steps below:
 - a) Generate a Certificate Signing Request (CSR) and send it to an Issuer (i.e. your trading partner or a Certificate Authority (CA) like Verisign).
 - b) The Issuer will sign your Certificate and will send you a CA reply.
 - c) If the CA reply does not contain the certificate chain of the Issuer, then you must import this chain.
 - d) Import the CA reply into your key store.
4. If your certificate was not signed by a third party (self-signed), then Export your Certificate and send it to your trading partner. In other words, the trading partner will trust your certificate, even though it was not signed by a CA.

Your Trading Partner's Certificate

Follow these steps if you want to use your trading partner's certificate for authenticating their server:

1. Request the certificate from your trading partner.
2. Import their certificate into the Trusted Certificates Key Store.
3. If this is a certificate chain, where the certificate that signed their certificate was signed by another certificate and so on, then all certificates in the chain must be added to your Key Store.

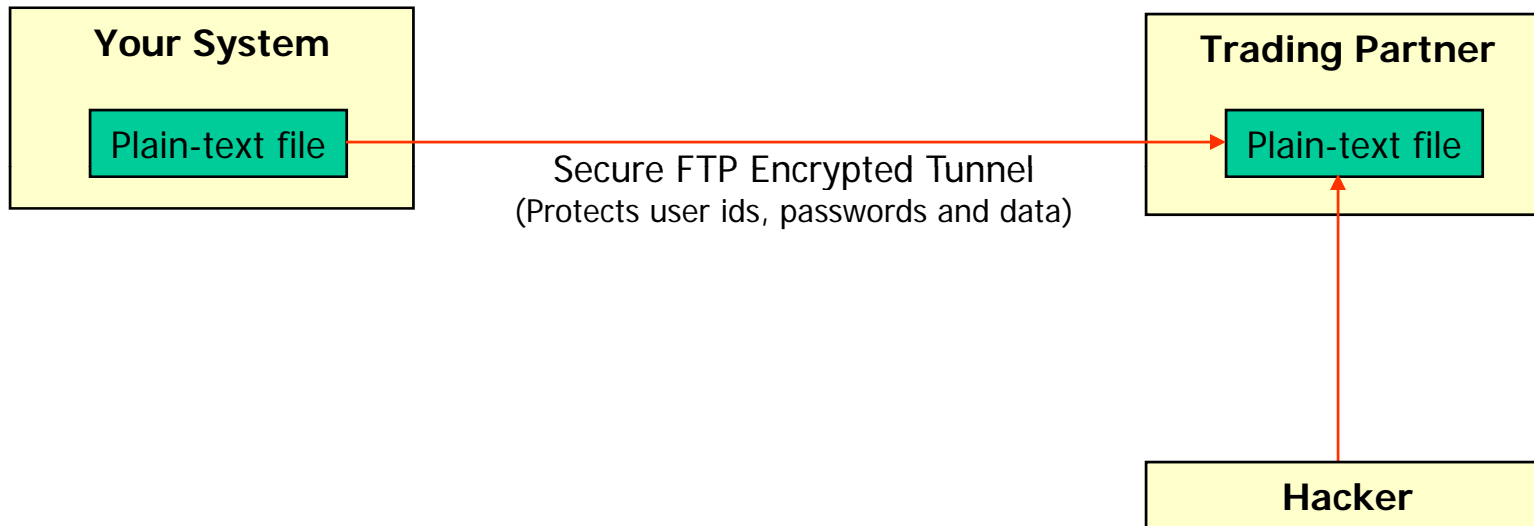
Should I use SFTP or FTPS?

- Both standards offer a high level of data protection
- Mainly depends on trading partner's capabilities
- SFTP is simpler to implement authentication (just use a key or password)
- SFTP is very popular in the LINUX/UNIX world
- SFTP uses the single port 22 for all communications, which is easy to open on your firewall
- FTPS can use signed certificates (3rd party verification) to establish a chain of trust, in which you don't need certificates from each partner
- FTPS often uses a random set of port numbers for each listing and data connection, which your firewall may block (you may have to set up a range of allowable port numbers in your firewall and configure in client)

Secure FTP – Potential issue

Secure FTP does not protect data “at rest”

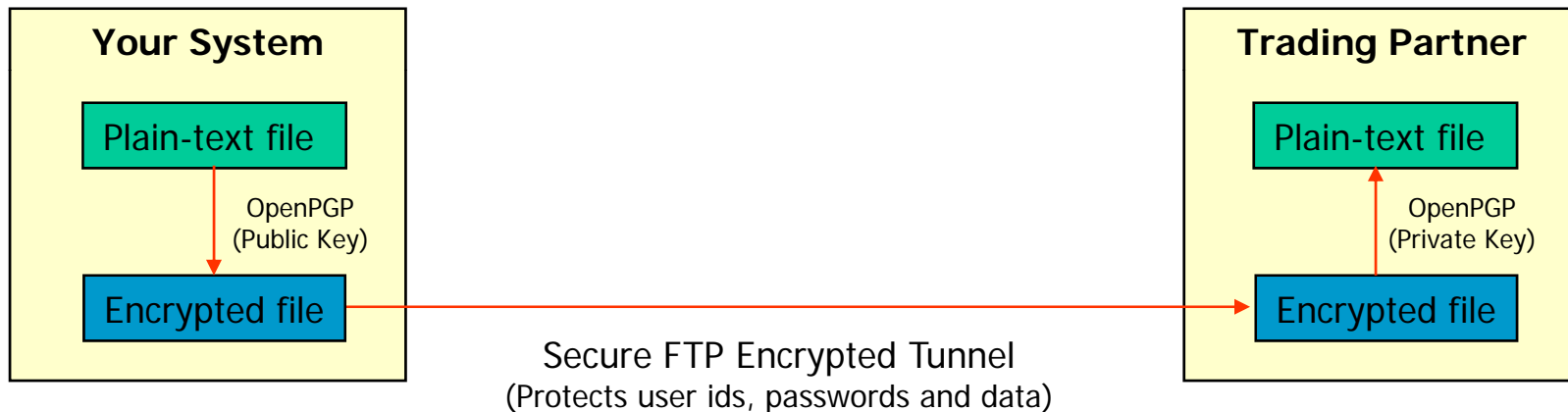
Is your trading partner protecting your data after it arrives at their site?



Secure FTP and OpenPGP – A One, Two Punch!

Secure FTP will protect your FTP user ids, passwords, commands and data while in transit

OpenPGP will also protect your data in transit, but also “at rest”



Note: Assumes that trading partner will decrypt your file only after it is moved to their internal protected network.

FTP Scripting and Management Issues

- Programmers traditionally have to write Scripts, CL programs, .BAT files, etc.
- Passwords are often stored in the scripts (in the clear)
- Scripts need to be maintained by Programmers:
 - When host names and IP addresses change
 - When user ids and passwords change
 - When file names change
- Can become very complex and unmanageable, especially as you add more trading partners
- Difficult to include/exclude files based on variables, wildcards, timestamps, sizes, etc.
- Lacking IF/ELSE controls (commands typically just run in sequential order)
- Some organizations use PC Tools for FTP/SFTP/FTPS transfers, in which files have to first be downloaded from corporate server:
 - This exposes the file(s) on the PC (vulnerable to hackers)
 - Download is often done manually to PC, which consumes time and is subject to errors

```
open ftp.bank.com
user joe password
ascii
get /inbound/ack837.txt
get /inbound/ack763.txt
lcd /orderfiles
put ord7632.txt /outbound/
put ord9383.txt /outbound/
close
quit
```

Error Trapping and Notification Issues

- Difficult to find and analyze FTP logs to determine when problems occur
- Example portion of a traditional FTP log:

```
> cd testftp
250 "TESTFTP" is current library.
Enter an FTP subcommand.
> lcd testftp
Local working directory is TESTFTP
Enter an FTP subcommand.
> put demodemo
File DEMODEMO in library TESTFTP not found. ← How do you know it failed?
Enter an FTP subcommand.
> quit
221 QUIT subcommand received.
```

- Manually or programmatically read through the OUTPUT log to find errors
- Are you notified when problems occur... or do you wait for the trading partner to call you?



L I N O M A
S O F T W A R E
REVOLUTIONARY SOLUTIONS FOR YOUR WORLD

Product Summary



GO ANYWHERE Director

GoAnywhere Director[™] streamlines and manages data movement through an innovative centralized approach. It allows your organization to connect to almost any system (internal or external) and securely exchange data using a wide variety of standard protocols.

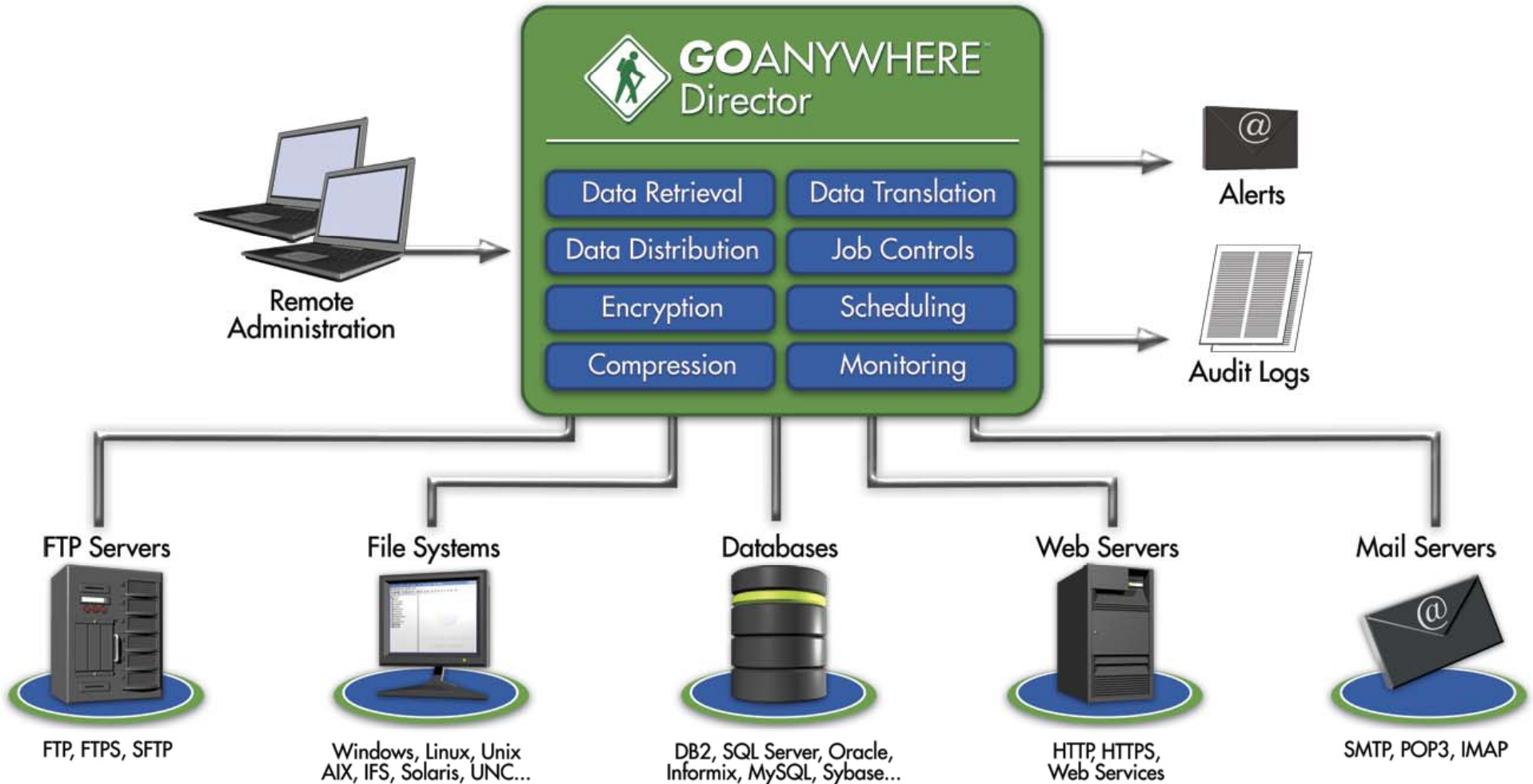


GO ANYWHERE Services

GoAnywhere Services[™] allows trading partners and employees to securely connect to your organization and easily retrieve or upload files. Popular file transfer and encryption standards are supported without the need for proprietary client software.



Product Diagram





- Moves data throughout the Enterprise
- Streamlines data transmissions with Trading Partners
- Eliminates or minimizes:
 - Custom programming traditionally needed
 - Manual processes
 - System Administration costs
 - Single function tools and PC products
 - Inefficient and costly VANs and Dial-Ups
- Secures data sent over the Network and Internet (SFTP, FTPS, SCP, OpenPGP, ZIP with AES)
- Decreases transfer times through compression (ZIP, GZIP, TAR)
- Translates data between popular formats (XML, Excel, CSV, Delimited Text, Fixed-Width Text)
- Provides centralized point-of-control and administration
- Includes detailed logging and message alerts
- Implements industry standards



Includes integrated Key Management tools for OpenPGP keys, SSL Certificates and SSH Keys.



GO ANYWHERE Director

Introduction

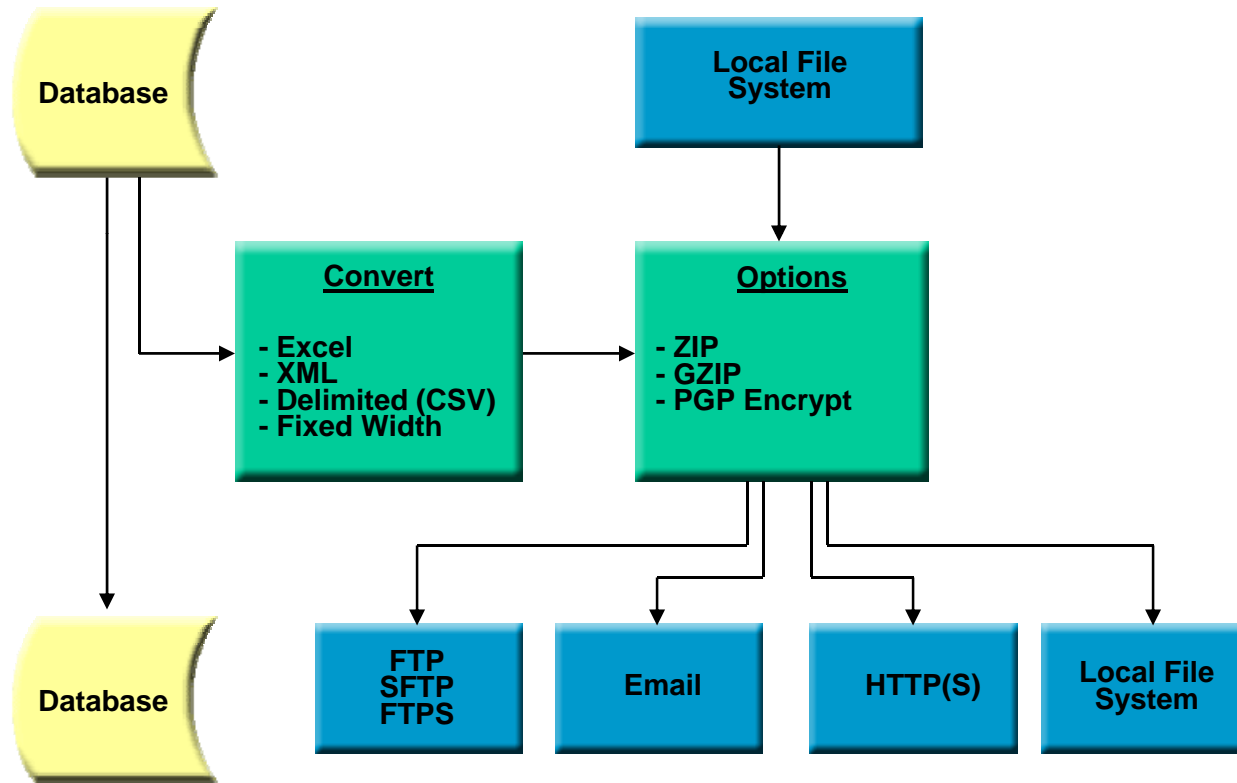
- Customer installable – Less than 30 minutes typically
- Includes over 50 different business processes (Tasks)
- Project-based design allows “chaining” of multiple Tasks together
- Automate Projects with built-in scheduler
- Launch Projects from other platforms, applications and programming languages



- Database
 - SQL
- Data Translation
 - Read CSV
 - Read Excel
 - Read Fixed-width
 - Read XML
 - Write CSV
 - Write Excel
 - Write Fixed-width
 - Write XML
- File Encryption
 - PGP Decrypt
 - PGP Encrypt
 - PGP Sign
 - PGP Verify
 - Unzip
 - Zip
- Compression
 - GUnzip
 - GZip
 - Unzip
 - Zip
- FTP
 - FTP
 - FTPS
 - SFTP
- E-Mail
- HTTP
- Local File System Tasks
- Native Calls
- Miscellaneous

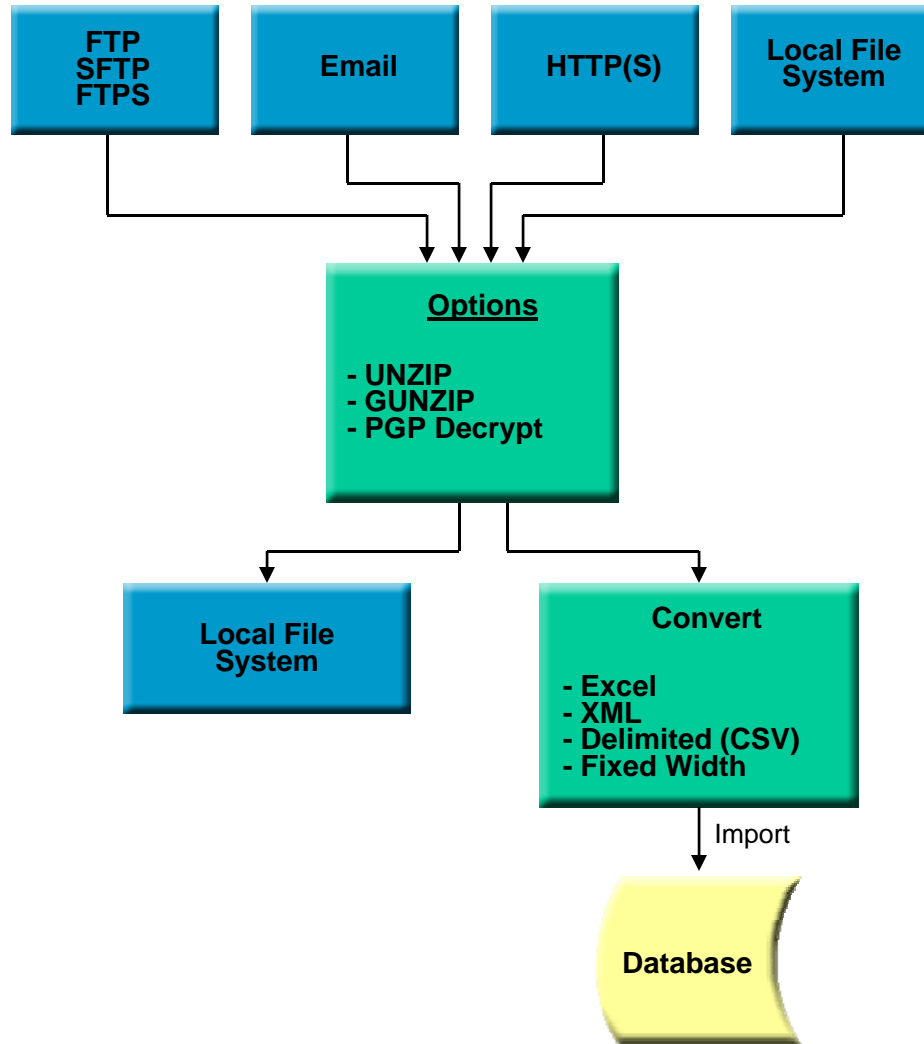


Automated Transfers – Outgoing



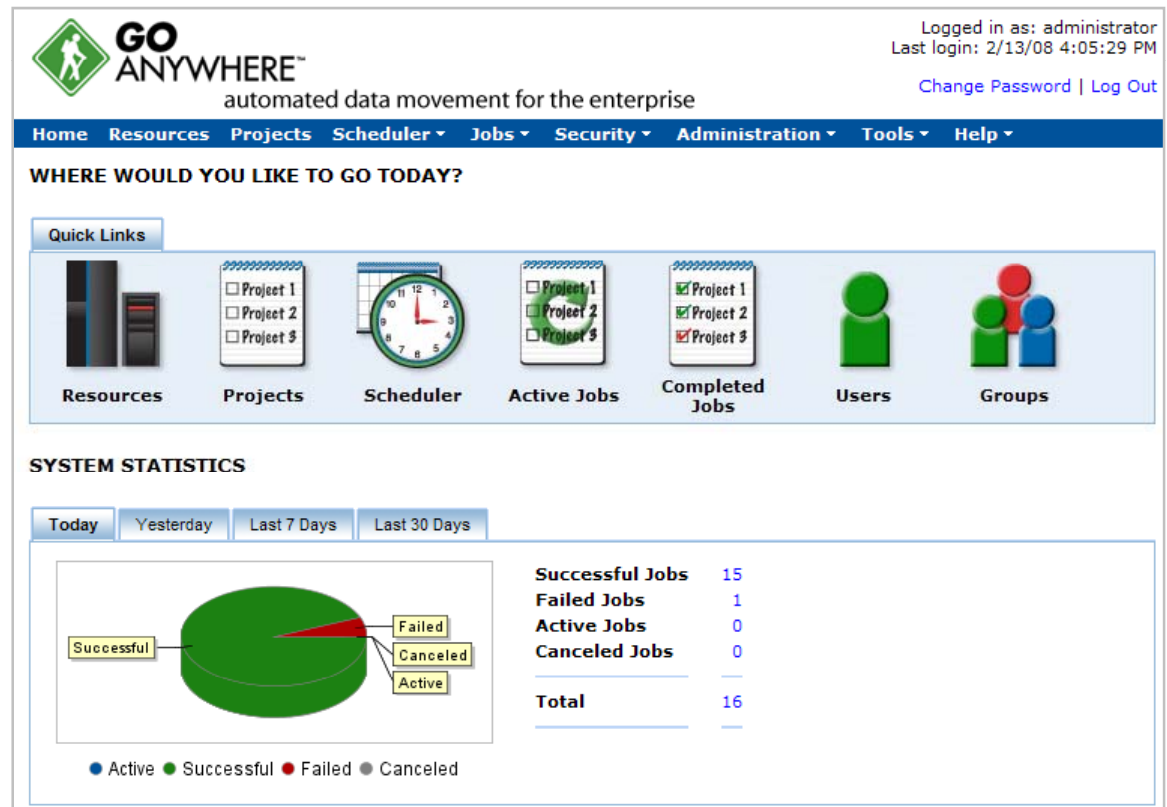


Automated Transfers – Incoming





- Browser based front-end for configuring GoAnywhere Projects
- Intuitive screens and wizards - AJAX enabled
- All definitions stored on server
- Graphical dashboard
 - Define Resources and Projects
 - Schedule and execute Projects
 - Monitor active Jobs
 - View Job Logs
 - Configure User Authority
 - View statistics





```
0001.00 PGM
0002.00 DCL &MESSAGE *CHAR 80
0003.00
0004.00 /* Run the Transfer */
0005.00 RUNPROJECT PROJECT ('/Payroll/SendDirectDeposit') +
0006.00         USER(SFIELD) PASSWORD(*****) +
0007.00         VARIABLE((StateCode NE)) PRIORITY(5)
0008.00
0009.00 /* Project failed or could not connect */
0010.00 MONMSG MSGID(GAE1002 GAE1003) EXEC(DO)
0011.00
0012.00 /* Get the error message from the program
0013.00 message queue */
0014.00 RCVMSG RMV(*NO) MSG(&MESSAGE)
0015.00
0016.00 /* Send the error to QSYSOPR */
0017.00 SNDMSG MSG(&MESSAGE) TOMSGQ(QSYSOPR)
0018.00 ENDDO
```

- Monitor for message ids
- Any errors are placed in job log
- Retrieve any errors with RCVMSG command



```
1/21/08 9:38:07AM INFO Start Date and Time: 1/21/08 9:38:07 AM
1/21/08 9:38:07AM INFO Job Number: 1200325835858
1/21/08 9:38:07AM INFO Project Name: /Demo/DB to Excel to Zip and FTP
1/21/08 9:38:07AM INFO Submitted By: administrator

1/21/08 9:38:07AM INFO Executing task 'Retrieve Records'
1/21/08 9:38:07AM INFO Executing statement select * from LIBRARY.EMP
1/21/08 9:38:08AM INFO Query execution produced a rowset
1/21/08 9:38:08AM INFO Finished task 'Retrieve Records'

1/21/08 9:38:08AM INFO Executing task 'Create Excel File'
1/21/08 9:38:09AM INFO 8 record(s) were written
1/21/08 9:38:09AM INFO Finished task 'Create Excel File'

1/21/08 9:38:09AM INFO Executing task 'Create ZIP File'
1/21/08 9:38:09AM INFO Compressing file '/files/employees.xls'
1/21/08 9:38:09AM INFO Number of files compressed: 1
1/21/08 9:38:09AM INFO Finished task 'Create ZIP File'

1/21/08 9:38:09AM INFO Executing task 'FTP the ZIP File'
1/21/08 9:38:09AM INFO Connecting to '192.168.1.2' at port '21'
1/21/08 9:38:10AM INFO Executing sub-task 'put'
1/21/08 9:38:10AM INFO Setting the data type to AUTO
1/21/08 9:38:10AM INFO Uploading '/files/employees.zip'
1/21/08 9:38:12AM INFO 1 file(s) were uploaded successfully
1/21/08 9:38:12AM INFO Finished sub-task 'put'
1/21/08 9:38:12AM INFO Closed the FTP connection
1/21/08 9:38:12AM INFO Finished task 'FTP the ZIP File'

1/21/08 9:38:12AM INFO Finished module 'main'
1/21/08 9:38:12AM INFO Finished project 'DB to Excel to Zip and FTP'
1/21/08 9:38:12AM INFO End Date and Time: 1/21/08 9:38:12 AM
```



GO ANYWHERE™ Services



L I N O M A

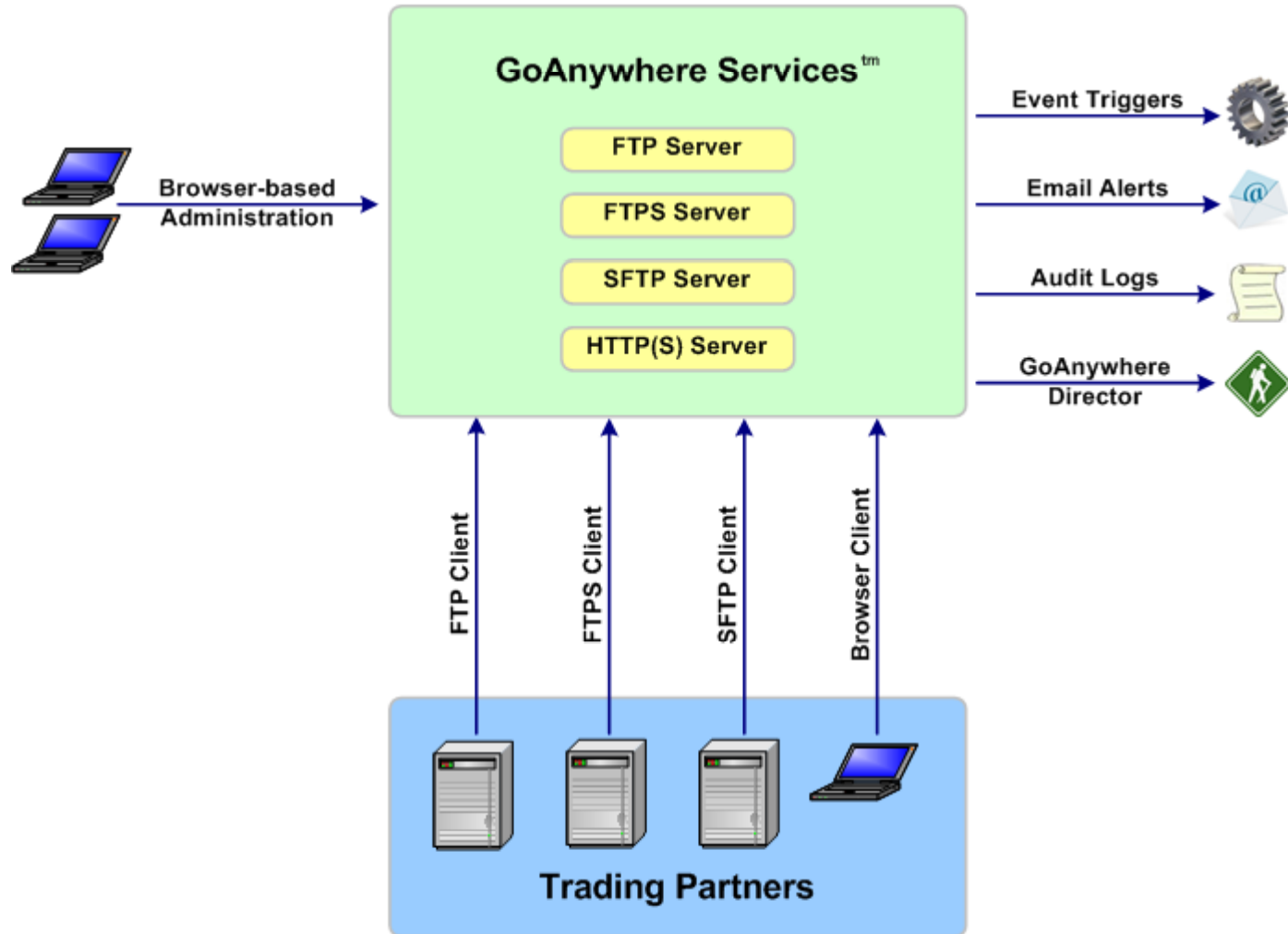
S O F T W A R E

REVOLUTIONARY SOLUTIONS FOR YOUR WORLD



GO ANYWHERE
Services

Diagram





- Allows your trading partners and employees to securely connect to your organization and easily retrieve or upload files.
- Supports open transfer protocols of FTP, SFTP, FTPS, HTTP/s
- Can secure transmissions with SSL/TLS or SSH encryption
- Provides a pure web client for simple file transfers
- Includes event triggers based on user-defined conditions
- Generates detailed audit logs and alert messages
- Provides trading partner account wizards and permission controls
- Intuitive browser-based interface for administration and monitoring
- No programming or special skills needed
- Installs to AIX, HP-UX, IBM System i, IBM System p, IBM System z, Linux, Microsoft Windows, Sun Solaris and UNIX





- Unlimited number of trading partners can be configured
- Grant individual permissions or adopt permissions from groups
- Restrict access based on the type of service (FTP, SFTP, FTPS, HTTP/s)
- Restrict access to certain functions (e.g. upload, download, delete, rename, etc.)
- Automatically send email with user id and password

EDIT WEB USER [?](#)

General Password Authentication Permissions and Groups

User Name

First Name

Last Name

Description

Organization

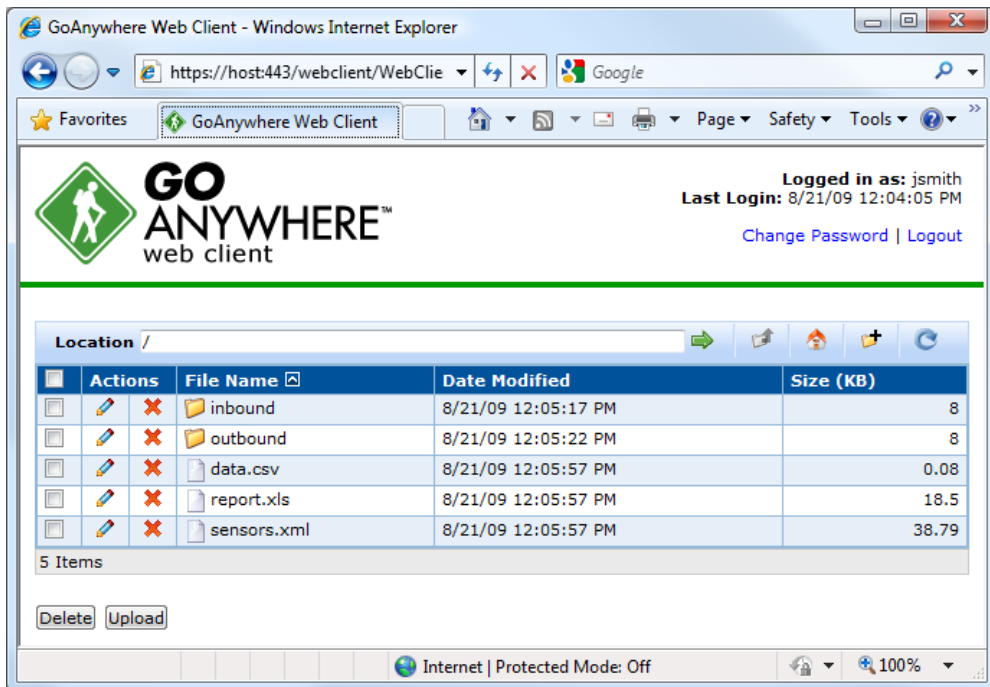
E-Mail Address

Phone

Account Expires On

Enabled

- Allows your trading partners simple access to your system for exchanging files
- Authenticate with user-ids/passwords/certificates with granular permission controls
- Full audit trails and event triggers
- Rebrand with your company logo



GoAnywhere Web Client - Windows Internet Explorer











https://host:443/webclient/WebClie

GoAnywhere Web Client

GO ANYWHERE
web client

Logged in as: jsmith
Last Login: 8/21/09 12:04:05 PM
[Change Password](#) | [Logout](#)

Location /

Actions	File Name	Date Modified	Size (KB)
 	inbound	8/21/09 12:05:17 PM	8
 	outbound	8/21/09 12:05:22 PM	8
 	data.csv	8/21/09 12:05:57 PM	0.08
 	report.xls	8/21/09 12:05:57 PM	18.5
 	sensors.xml	8/21/09 12:05:57 PM	38.79

5 Items

Delete Upload

Internet | Protected Mode: Off 100%



- GoAnywhere Director can send/retrieve files to/from GoAnywhere Services
- GoAnywhere Services can call Projects in GoAnywhere Director based on triggers
- Triggers based on file upload, download, rename, etc.
- Pass parameters, such as user and file name
- Can run multiple triggers per event

TRIGGER DETAILS [?](#)

Name	Decrypt File
Description	
Event Type	Upload Successful
Status	Active
Priority	2
Stop Processing More Triggers	Yes
Service	<input checked="" type="checkbox"/> HTTPS <input checked="" type="checkbox"/> FTP <input checked="" type="checkbox"/> FTPS <input checked="" type="checkbox"/> SFTP
Created By	sluebbe
Created On	8/31/09 10:51:17 AM
Modified By	sluebbe
Modified On	8/31/09 10:51:17 AM

Conditions

Match Type	All
-------------------	-----

Attribute	Expression	Comparison Value
event.userName	Equals	sluebbe

Action

Action Type	Call GoAnywhere Project
Resource	GoAnywhere Demo
Override User	
Override Password	*****
Project	/Steve projects/services/Decrypt

Variables

Name	Value
filePath	\${event.physicalPath}

- Audit logs stored for every transaction (login, upload, download, rename, etc.) for all services
- Search using a wide variety of filter criteria
- View on-line or export to CSV

FTP AUDIT LOG ?

Basic Search **Advanced Search**

Date Range 9/1/09 12:00 AM to 9/18/09 12:00 AM

User **Severity*** Info Warning Error

Event ID

Command* Login Download Upload Mkdir Change Password
 Logout Delete Rename Chmod Connect
 Disconnect

Local IP **Remote IP**

Session ID

File Path Equals

Results Per Page 20 [Show/Hide Columns](#)

Showing from 1 through 19

Actions	Start Time	Time(ms)	Command	Severity	User	File Path
	9/1/09 4:44:05 PM	108	Connect			
	9/1/09 4:44:05 PM	3	Login		sluebbe	
	9/1/09 4:44:15 PM	7	Rename		sluebbe	/linoma/gaservices/userdata/webdocs/sluebbe/2009_NCAA_F-B-S_Schedule.xls
	9/1/09 4:44:18 PM	68	Delete		sluebbe	/linoma/gaservices/userdata/webdocs/sluebbe/2009.xls
	9/1/09 4:44:21 PM	24	Connect			
	9/1/09 4:44:21 PM	7	Login		sluebbe	
	9/1/09 4:44:21 PM	4344	Upload		sluebbe	/linoma/gaservices/userdata/webdocs/sluebbe/2009_NCAA_F-B-S_Schedule.xls
	9/1/09 4:45:25 PM	1	Disconnect		sluebbe	

Installation Requirements

System i:

- | | |
|---------------------------|----------------|
| - Operating System | V5R3 or higher |
| - JVM | 1.5 |
| - Disk space requirements | 75 MB |
| - Memory requirements | 256 MB |

Windows:

- | | |
|---------------------------|-------------------------------------|
| - Operating System | Windows 2000, 2003, 2008, XP, Vista |
| - JVM | 1.5 (installed with product) |
| - Disk space requirements | 125 MB |
| - Memory requirements | 256 MB |

UNIX / Linux / AIX / Solaris / HP-UX:

- | | |
|---------------------------|---------------------------|
| - Operating System | LINUX or UNIX |
| - JVM | 1.5 (installed for Linux) |
| - Disk space requirements | 125 MB |
| - Memory requirements | 256 MB |



L I N O M A

S O F T W A R E

REVOLUTIONARY SOLUTIONS FOR YOUR WORLD

How to Contact Us

Web site: www.GoAnywhereMFT.com

E-mail: sales@linomasoftware.com

Toll-free: 1-800-949-4696

Direct: (402) 944-4242

Fax: (402) 944-4243

Address: 1409 Silver Street
Ashland, NE 68003 USA