Global Information Assurance Certification Paper

SANS Security Essentials
GSEC Practical Assignment
Version 1.2c Current as of December, 2000 (amended April 16, 2001)

SSH And Local Tunnels – Encrypting User Defined Ports
Phil Kramer
April 17, 2001

Why I needed to do this

Recently a client called me and said that he thought that he might need another firewall. He is in an educational environment where students, faculty and the school's commercial entities are intermixed on the same network. Much of this network is hub and router based, not switched. After several minutes of discussion we finally came to the crux of his problem; the school had written its own client/server point of sale application so they could accept credit card payments in their commercial entities on campus and recently some of the credit cards that had been used on campus have had additional charges show up on these credit cards. The school is certain that the credit card information was being captured somewhere on their network.

I discussed with him that the NT server and any WIN client workstations that are involved in the point of sale transaction must locked down to rule out access at the clients and server. I asked were they using strong passwords, had they removed shares, updated registry entries for null user, etc. as per the "SANS Institute's Windows NT Security Step By Step" guide. He said that the server and workstations were locked down and they had not been compromised. I then asked if the transactions going across the LAN were encrypted and he said no they were not.

This is where I started to think about encryption solutions such as VPN and told the client I would get back with him. I hung up the phone and turned back to finish the email I was working. After sending the email, I was just looking at my desktop when I saw it, my icon for "F-Secure SSH Tunnel and Terminal". Right then and there I knew I had a solution. I reasoned that I did not want to encrypt all the data that would be entering and leaving the workstations and server, I only wanted to encrypt the information for the port that the point of sale application would be using. I then set this up in my lab to see how it would work.

There are 3 GSEC papers on SSH that cover the internals, installation and configuration of SSH Terminal on Windows and Unix so please see the References Section for more information on these topics.

SSH Local Tunnel Server Configuration Information:

The only configuration change on the server side is that TCP tunneling must be enabled. The following is the server screen where the "Allow TCP tunneling" must be enabled and can optionally be further secured.

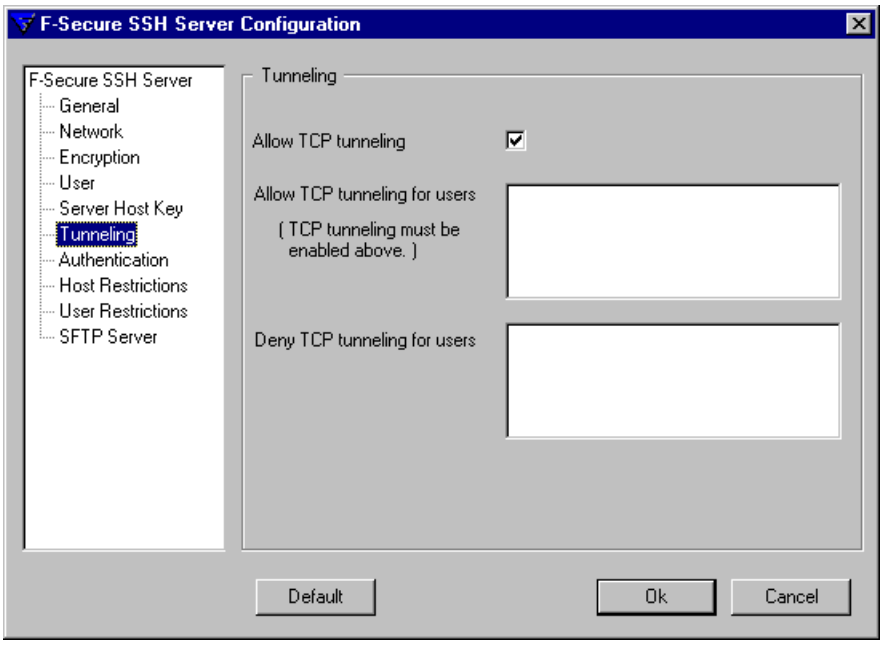

Figure 1.

Non-SSH Local Tunnel Client Configuration Information:
The only configuration issue I came upon outside of the SSH software was on the clients. The client software requires that a valid host entry for the local loopback address exists in the local hosts file. The following is an example of this entry:
127.0.0.1 localhost

A previous install of Netscape Navigator mucked up my localhost entry. The reason you will need the localhost entry on the clients will be expanded on later.

Depending on your OS the hosts file are normally found in:
WIN9X systemroot/WINDOWS/hosts
NT4 systemroot/WINNT/system32/drivers/etc/hosts
*NIX /etc/hosts

SSH Local Tunnel Client Configuration Information:
In addition to the normal configuration information needed for an SSH client Terminal connection to an SSH server, the local tunnel information must be entered. This information consists of a source port, destination host and destination port rule for each local tunnel.

NOTE - The local tunnel information can be added dynamically after the connection to the server is established, but the local tunnel information cannot be edited or deleted while a connection is established.

NOTE – This note pertains to *NIX SSH servers only. Only high numbered ports can be locally tunneled if the SSH daemon is not started by root. On *NIX systems only root can listen on low numbered ports.

The following is the normal SSH client configuration information needed to connect to a SSH server 192.168.0.2. I optionally enabled compression.

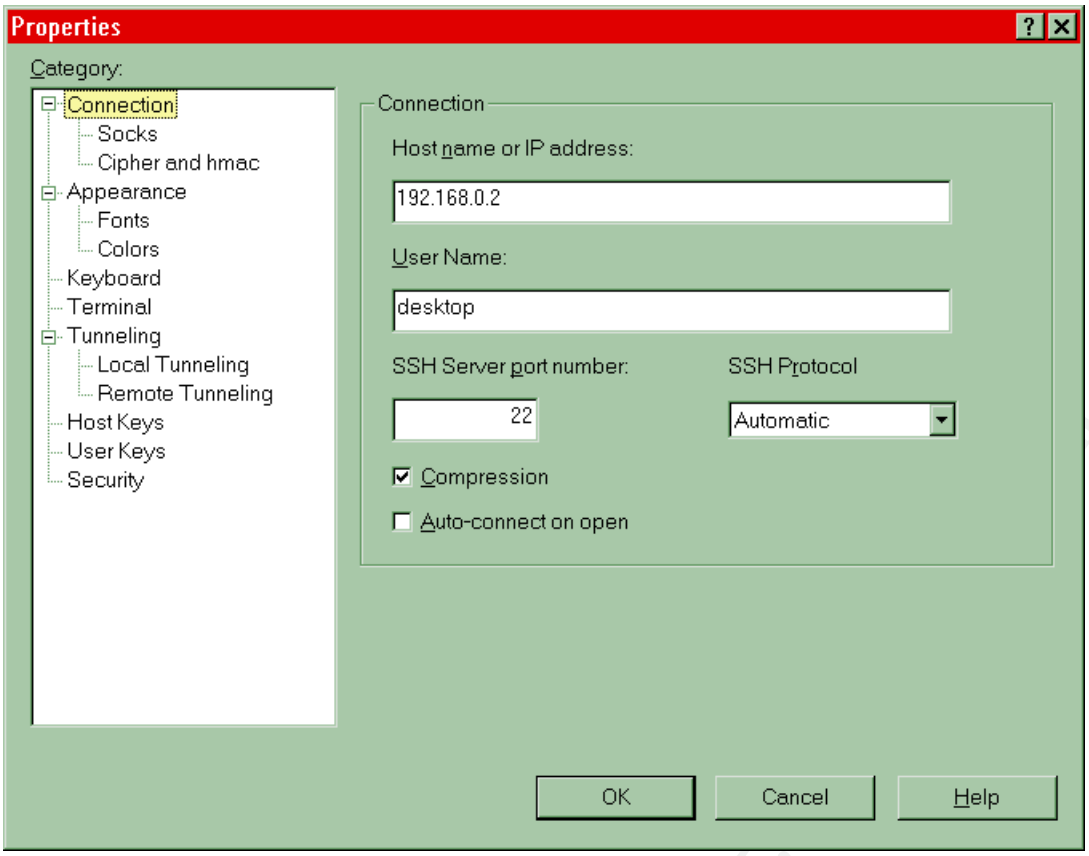

Figure 2.

The following is the additional configuration needed to create a local tunnel between the SSH client and an SSH server.

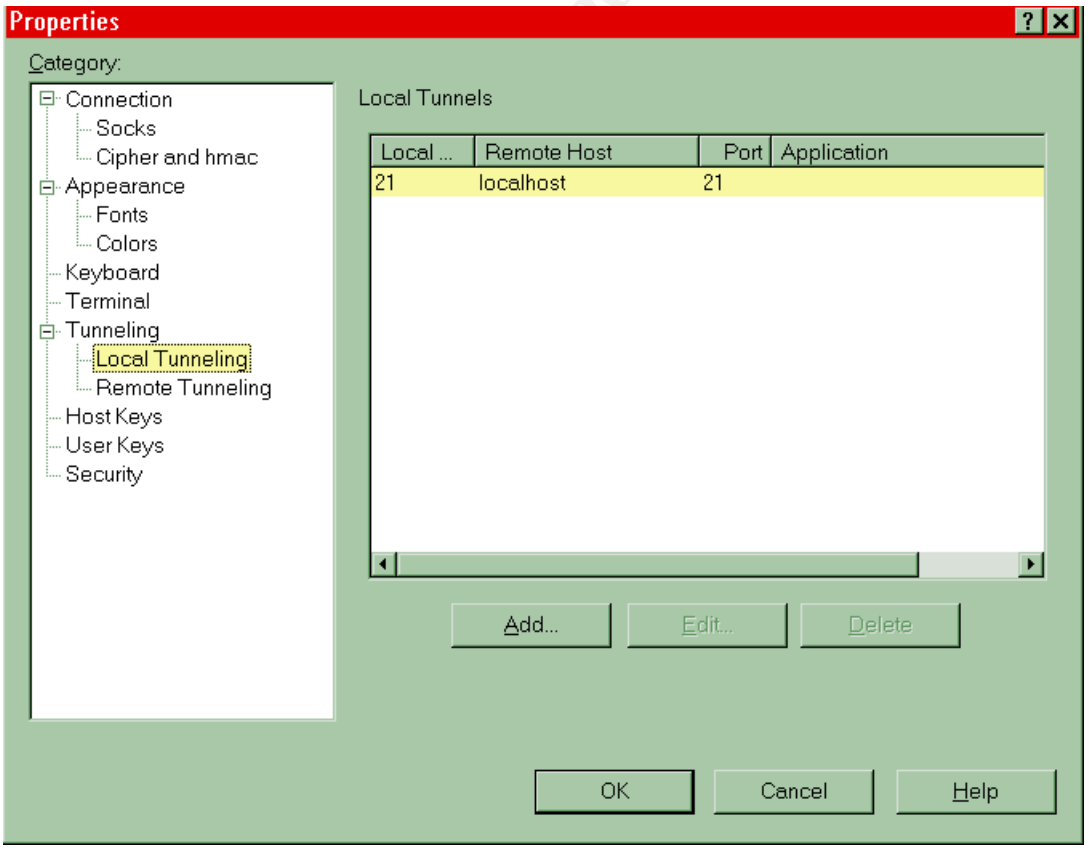

Figure 3.

As you can see I have set up the local source port 21 be mapped to destination port 21. This does not always have to be the case but for this example I am using tcp/21 for FTP control information and my client and server FTP applications do not allow for the changing of the port to be used.

The one peculiar thing about this configuration that should stand out is that the remote host that is entered into the local tunnel configuration is NOT a remote host. It is for the host, localhost, whose IP address is in the hosts file as 127.0.0.1. We could have named this anything we wanted as long as the IP address is 127.0.0.1 and we use this host name when invoking the application for its remote connection. The reason you will need the localhost entry on the clients will be expanded on later.

Figure 4 below shows the output of the netstat command, "netstat –an", of a client workstation prior to running either the SSH client or a FTP client. You can see that there is not much network activity going on besides the client machine, whose network address is 192.168.0.1, is listening on Window's ports 137, 138 and 139.

```
Active Connections
  Proto  Local Address    Foreign Address State
  TCP    192.168.0.1:137  0.0.0.0:0       LISTENING
  TCP    192.168.0.1:138  0.0.0.0:0       LISTENING
  TCP    192.168.0.1:139  0.0.0.0:0       LISTENING
  UDP    192.168.0.1:137  *:*
  UDP    192.168.0.1:138  *:*
```

Figure 4.

NOTE – For the sake of clarity I will remove the entries for ports 137-139 from the rest of the netstat outputs.

Figure 5 below shows the output of the netstat command, "netstat –an", of an FTP client workstation connecting to an FTP server. You can see we now have two new entries in our table:

Entry 1 shows an FTP client workstation, whose default network address is 0.0.0.0, listening on port 1035.

Entry 2 shows an established connection between an FTP client workstation, whose network address is 192.168.0.1, using high port 1035 and the FTP server, whose network address is 192.168.0.2, using the normal FTP server port 21.

```
Active Connections
  Proto  Local Address    Foreign Address State
  TCP    0.0.0.0:1035     0.0.0.0:0       LISTENING   ← 1
  TCP    192.168.0.1:1035 192.168.0.2:21  ESTABLISHED ← 2
```

Figure 5.

Figure 6 below shows the output of the netstat command, "netstat –an" of an SSH client workstation connecting to an SSH server. These entries are exactly the same as the FTP entries listed above with the only exceptions being that the local port is now 1193 and the destination port is now 22. You can see the two different entries in the table:

Entry 1 is the local SSH client workstation, whose default network address is 0.0.0.0, listening on port 1193.

Entry 2 is an established connection between an SSH client workstation, whose network address is 192.168.0.1, using high port 1193 and the SSH server, whose network address is 192.168.0.2, using the normal SSH port 22.

```
Active Connections

  Proto  Local Address    Foreign Address State
  TCP    0.0.0.0:1193     0.0.0.0:0       LISTENING   ← 1
  TCP    192.168.0.1:1193 192.168.0.2:22  ESTABLISHED ← 2
```

Figure 6.

The figure below shows the output of the netstat command, "netstat –an", of an SSH client workstation connecting to an SSH server with a local tunnel defined. The definition for the local tunnel is the following:
source port = 21
remote host = localhost, localhost's IP address is 127.0.0.1 from the client's "hosts" file.
destination port - 21

Entry 1 is the local SSH client workstation, whose default network address is 0.0.0.0, listening on port 1193. This is the same entry as 1 above.

Entry 2 is the local tunnel definition created on the SSH client workstation, whose network address is the local loopback address is 127.0.0.1, listening on port 21.

Entry 3 is an established connection between an SSH client workstation, whose network address is 192.168.0.1, using high port 1193 and the SSH server, whose network address is 192.168.0.2, using the normal SSH port 22. This is the same entry as 2 above.

NOTE - Since I have not logged into the FTP server I do not have any established sessions that use port 21. The localhost 127.0.0.1 is only listening on port 21 at this point.

```
Active Connections
  Proto  Local Address    Foreign Address State
  TCP    0.0.0.0:1193     0.0.0.0:0       LISTENING   ← 1 Same as 1 above
  TCP    127.0.0.1:21     0.0.0.0:0       LISTENING   ← 2
  TCP    192.168.0.1:1193 192.168.0.2:22  ESTABLISHED ← 3 Same as 2 above
```

Figure 7.

The figure below shows the output of the netstat command, "netstat –an", of an SSH client workstation connecting to an SSH server with a local tunnel defined and FTP session active.

Entry 1 is the local SSH client workstation, whose default network address is 0.0.0.0, listening on port 1193. This is the same entry as 1 above.

Entry 2 is the local tunnel definition created on the SSH client workstation, whose network address is the local loopback address is 127.0.0.1, listening on port 21. This is the same entry as 2 above.

Entry 3 is one half of the local tunnel, localhost 127.0.0.1 listening on port 21 has an established connection with localhost 127.0.0.1 listening on port 1195.

Entry 4 is the other half of the local tunnel localhost 127.0.0.1 listening on port 1193 has an established connection with localhost 127.0.0.1 listening on port 21.

Entry 5 is an established connection between an SSH client workstation, whose network address is 192.168.0.1, using high port 1193 and the SSH server, whose network address is 192.168.0.2, using the normal SSH port 22. This is the same entry as 3 above.

```
Active Connections
  Proto  Local Address    Foreign Address State
  TCP    0.0.0.0:1193     0.0.0.0:0       LISTENING   ← 1 Same as 1 above
  TCP    127.0.0.1:21     0.0.0.0:0       LISTENING   ← 2 Same as 2 above
  TCP    127.0.0.1:21     127.0.0.1:1195  ESTABLISHED ← 3 Half of tunnel
  TCP    127.0.0.1:1195   127.0.0.1:21    ESTABLISHED ← 4 Half of tunnel
  TCP    192.168.0.1:1193 192.168.0.2:22  ESTABLISHED ← 5 Same as 3 above
```

Figure 8.

How To Invoke The Configured Local Tunnel

First we make a connection using our SSH client to connect to the SSH server just like we were making a secure terminal session connection. The following is the SSH Client Logon Information screen:

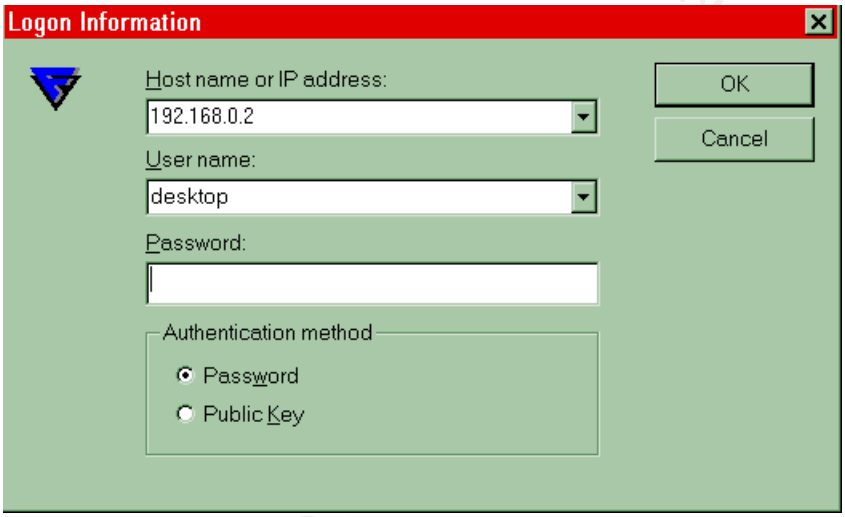

Figure 9.

After establishing an SSH Terminal connection to the SSH server we now start the FTP application from the client workstation by entering:
C:\>ftp localhost -or-
C:\>ftp 127.0.0.1

The FTP command must be invoked with a host that has an IP address of 127.0.0.1 in order for the local tunnel to be enabled. If the hosts file had the following entry:
127.0.0.1 dog

We could have connected using:
C:\>ftp dog

Any of the three above commands would start the client FTP session back to the client itself with the IP address of 127.0.0.1. SSH knows that it is supposed to encrypt and forward any port 21 packets destined for the server that has this source port defined in its local tunnel configuration and not to machine dog. Remember the local tunnel information is entered for an SSH connection to a specific SSH server.

Examples Of Why I Wanted To Do This In The First Place

I wanted to encrypt my client/server communications to ensure confidentiality. Here is a summary sniff of the FTP session without an SSH local tunnel. As you can see all the requests are in plain text, the USER "desktop" and PASS "desktop" specifically. FTP is definitely lacking in confidentiality.

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.1	192.168.0.2	TCP	1038 > 21 [SYN] Seq=934050 Ack=0 Win=5360 Len=0
2	0.000251	192.168.0.2	192.168.0.1	TCP	21 > 1038 [SYN, ACK] Seq=57068 Ack=934051 Win=8576 Len=0
3	0.000531	192.168.0.1	192.168.0.2	TCP	1038 > 21 [ACK] Seq=934051 Ack=57069 Win=5360 Len=0
4	0.007802	192.168.0.2	192.168.0.1	FTP	Response: 220-This FTP site is running a copy of WFTPD that
5	0.008087	192.168.0.2	192.168.0.1	FTP	Response: 20-to five (5) transfers - to get more transfers,
6	0.008313	192.168.0.1	192.168.0.2	TCP	1038 > 21 [ACK] Seq=934051 Ack=57953 Win=5360 Len=0
7	2.137473	192.168.0.1	192.168.0.2	FTP	**Request: USER desktop**
8	2.137924	192.168.0.2	192.168.0.1	FTP	Response: 331 Give me your password, please
9	2.292981	192.168.0.1	192.168.0.2	TCP	1038 > 21 [ACK] Seq=934065 Ack=57988 Win=5325 Len=0
10	4.554515	192.168.0.1	192.168.0.2	FTP	**Request: PASS desktop**
11	4.566607	192.168.0.2	192.168.0.1	FTP	Response: 230 Logged in successfully
12	4.697020	192.168.0.1	192.168.0.2	TCP	1038 > 21 [ACK] Seq=934079 Ack=58016 Win=5297 Len=0
13	7.677405	192.168.0.1	192.168.0.2	FTP	**Request: HELP**
14	7.678425	192.168.0.2	192.168.0.1	FTP	Response: 214-The following commands are recognized (* =>'s
15	7.812939	192.168.0.1	192.168.0.2	TCP	1038 > 21 [ACK] Seq=934085 Ack=58549 Win=5360 Len=0
16	9.213503	192.168.0.1	192.168.0.2	FTP	**Request: QUIT**
17	9.214259	192.168.0.2	192.168.0.1	FTP	Response: 221-This copy of WFTPD is STILL NOT REGISTERED
18	9.215848	192.168.0.2	192.168.0.1	TCP	21 > 1038 [FIN, ACK] Seq=58827 Ack=934091 Win=8536 Len=0
19	9.216081	192.168.0.1	192.168.0.2	TCP	1038 > 21 [ACK] Seq=934091 Ack=58828 Win=5082 Len=0
20	9.254313	192.168.0.1	192.168.0.2	TCP	1038 > 21 [FIN, ACK] Seq=934091 Ack=58828 Win=5082 Len=0
21	9.254476	192.168.0.2	192.168.0.1	TCP	21 > 1038 [ACK] Seq=58828 Ack=934092 Win=8536 Len=0

Figure 10.

Here is a summary sniff of the same FTP commands using SSH and local tunnels. As you can see the entire FTP session is transmitted back and forth as if it were the normal encrypted SSH terminal data. All data is encrypted including the FTP session's user name and password, high marks for confidentiality.

Note – There are fewer packets in this dump then there were in the one above. I presume that since port 22 is used for the FTP connection the packet count differs because of the lack of the three-way handshake and connection teardown. Since the data is encoded I cannot tell.

```
No. Time        Source                Destination           Protocol Info
  1 0.000000    192.168.0.1           192.168.0.2           TCP      1193 > 22 [PSH, ACK] Seq=1673783 Ack=60230 Win=5040 Len=36
  2 0.003921    192.168.0.2           192.168.0.1           TCP      22 > 1193 [PSH, ACK] Seq=60230 Ack=1673819 Win=8504 Len=36
  3 0.012582    192.168.0.2           192.168.0.1           TCP      22 > 1193 [PSH, ACK] Seq=60266 Ack=1673819 Win=8504 Len=44
  4 0.012918    192.168.0.1           192.168.0.2           TCP      1193 > 22 [ACK] Seq=1673819 Ack=60310 Win=4960 Len=0
  5 3.216787    192.168.0.1           192.168.0.2           TCP      1193 > 22 [PSH, ACK] Seq=1673819 Ack=60310 Win=4960 Len=36
  6 3.218688    192.168.0.2           192.168.0.1           TCP      22 > 1193 [PSH, ACK] Seq=60310 Ack=1673855 Win=8468 Len=36
  7 3.341896    192.168.0.1           192.168.0.2           TCP      1193 > 22 [ACK] Seq=1673855 Ack=60346 Win=4924 Len=0
  8 5.315909    192.168.0.1           192.168.0.2           TCP      1193 > 22 [PSH, ACK] Seq=1673855 Ack=60346 Win=4924 Len=36
  9 5.330401    192.168.0.2           192.168.0.1           TCP      22 > 1193 [PSH, ACK] Seq=60346 Ack=1673891 Win=8432 Len=36
 10 5.442065    192.168.0.1           192.168.0.2           TCP      1193 > 22 [ACK] Seq=1673891 Ack=60382 Win=4888 Len=0
 11 9.297115    192.168.0.1           192.168.0.2           TCP      1193 > 22 [PSH, ACK] Seq=1673891 Ack=60382 Win=4888 Len=44
 12 9.299933    192.168.0.2           192.168.0.1           TCP      22 > 1193 [PSH, ACK] Seq=60382 Ack=1673935 Win=8388 Len=324
 13 9.475862    192.168.0.1           192.168.0.2           TCP      1193 > 22 [ACK] Seq=1673935 Ack=60706 Win=5360 Len=0
 14 14.319513   192.168.0.1           192.168.0.2           TCP      1193 > 22 [PSH, ACK] Seq=1673935 Ack=60706 Win=5360 Len=36
 15 14.322932   192.168.0.2           192.168.0.1           TCP      22 > 1193 [PSH, ACK] Seq=60706 Ack=1673971 Win=8352 Len=44
 16 14.407522   192.168.0.1           192.168.0.2           TCP      1193 > 22 [PSH, ACK] Seq=1673971 Ack=60750 Win=5316 Len=36
 17 14.408517   192.168.0.2           192.168.0.1           TCP      22 > 1193 [PSH, ACK] Seq=60750 Ack=1674007 Win=8316 Len=36
 18 14.511811   192.168.0.1           192.168.0.2           TCP      1193 > 22 [ACK] Seq=1674007 Ack=60786 Win=5280 Len=0
```

Figure 11.

Conclusion

SSH can provide you with a low-cost, multi-platform, client/server software encryption package with the one caveat being that this is the only server on the network using this port that the client will need to communicate with. If you set up multiple server connections and each one is set up to use local tunnels on the same port, only the last server you connect with will have the local tunnels active. Any other local tunnels that are defined to go to other hosts using that same port are not enabled.

We should all be aware of any exploits or vulnerabilities that exist for any products that we are recommending or installing. At last count I found 33 advisories/vulnerability notes on Cert/CC pertaining to SSH. Most of the listed vulnerabilities apply to SSH version 1 or the RSA encryption weakness. The latest version of F-Secure's SSH I tested with is version 2.3.1 which does not contain the listed vulnerabilities and uses DSA encryption as the default.

Implementation - Security Note We should always test our implementations. I had incorrectly configured my setup and while I thought I was encrypting my FTP traffic on port 22, a quick "netstat –an" showed that my FTP session was actually connected on port 21. A sniff of my network further proved that I was not encrypting the traffic.

Products Used In This Paper

F-Secure's SSH Tunnel and Terminal Client for Win/NT/*NIX and SSH Server for Windows NT are commercial applications that may have additional functionality and features not found in the freeware/shareware versions. I have the SSH Server software residing on the same machine as the WFTPD software. This can be split up between 2 different machines, but that would be implementing "remote tunnels" which is beyond the scope of this paper. In addition, running FTP through a local tunnel would not be necessary in the real world because F-Secure's SSH Client and Server products have SFTP (Secure FTP) and SCP (Secure cp) applications built into them.

Texas Imperial Software's WFTPD is a freeware/share FTPd for WIN/NT. I used the WFTPD software so I would have a server application that would listen on ports other than the default SSH port and the X-Window port.

AnalogX's Capture is a freeware/share program for WIN/NT used for capturing screenshots.

Ethereal is a Win/NT/*NIX for network sniffing.

Netstat is a Win/NT/*NIX application that reports information about port activity and connection state on a machine.

List Of References

1. SSH, Secure Shell Chuck Crawford October 12, 2000
http://www.sans.org/infosecFAQ/authentic/SSH.htm

2. Using SSH2 for UNIX and Windows Paul Koppel November 25, 2000
http://www.sans.org/infosecFAQ/encryption/SSH2.htm

3. X Tunnels through SSH Layne Bro November 30, 2000
http://www.sans.org/infosecFAQ/encryption/x_tunnels.htm

4. AnalogX Capture a Win9X/NT program for screenshots

http://www.analogx.com

5. Ethereal a Win9x/NT/*NIX program for network sniffing
http://www.ethereal.com

6. Texas Imperial Software WFTPD a Win9x/NT FTP server program
http://www.wFTPd.com

7. Cert/CC The CERT Coordination Center
http://www.cert.org