Interested in learning more about security?

SANS Institute InfoSec Reading Room

Configuring Secure Shell with TCP Wrappers on Solaris 2.8

Using Secure Shell (SSH) is one way to tighten the system. It encrypts all data transmissions, including login id and password, whether through the Internet or local network. It is a "best practice" to use Secure Shell on the local network as well because one cannot trust all the users within the network. The free version of Secure Shell is called OpenSSH. OpenSSH is widely used, and is available for various platforms. The OpenSSH product replaces the telnet and rlogin with ssh, replaces rcp with scp, ftp with sftp. Th...

Configuring Secure Shell with TCP Wrappers on Solaris 2.8

By Jane Micheller
August 8, 2002

GSEC Practical Assignment v1.4

Table of Contents

Introduction

The common trend today among federal, state, and local government, corporations, financial institutions, and companies of all sizes is dependence on the Internet for sending and receiving data. This reliance poses significant threats from hackers, unsecured individual workstations, worms, and viruses. Operating systems like Solaris, HP-UX, Linux, Microsoft, and others are not designed with security in-mind, or pre-loaded with firewall, anti-virus, or other products to help prevent systems from being hacked. A system administrator, a person who is expert on the operating system, must customize the system to protect against malicious users. He/she must implement third party products to secure the operating system. If the system allows remote access via telnet, rsh, rcp, ftp, or rlogin, these utilities, transmit plain text across the network. An amateur hacker can install a "sniffer" program that can listen for any data that transmitted from any computer on the network[1]. The hacker can easily obtain login information and other confidential data that can be used to compromise the system.

Third party software is needed to help prevent "sniffing" and to discourage the hackers from tampering with the system. There are a number of ways to tighten the operating system. Using Secure Shell (SSH) is one way to tighten the system. It encrypts all data transmissions, including login id and password, whether through the Internet or local network. It is a "best practice" to use Secure Shell on the local network as well because one cannot trust all the users within the network. The free version of Secure Shell is called OpenSSH. OpenSSH is widely used, and is available for various platforms. The OpenSSH product replaces the telnet and rlogin with ssh, replaces rcp with scp, ftp with sftp. This paper shows how to setup the OpenSSH version 3.4 on Solaris 2.8 platform, beginning with the development of the product and illustrates packet captures.

History of SSH

The first version of Secure Shell (SSH) version 1.2.12 protocol was written by Tatu Ylönen and released on July 1995[2]. Björn Grönvall re-discovered this release in 1999, and started fixing bugs[3]. The OpenBSD project group found his work, and wanted to include the SSH protocol support in the OpenBSD 2.6 release. Therefore, the OpenSSH is a derivative from this original free version and was re-created by six people in OpenBSD project group: Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt, and Dug Song. They have fixed many bugs, added newer features like OpenSSL, Keberos authentication and ticket passing, and one-time password authentication with s-key. However, they retained the mathematical functions from libgmp libary as in the original version of SSH.

Examination of Telnet and SSH Packet Capture

If a hacker uses sniffer or eavesdrop utilities on a remote machine, he/she will be able to see the user **Telnet** session**.** Illustrated below is an example using “snoop”, a UNIX

command to capture data transmissions. Any thing the user types is broadcast over the wire in plain text, as illustrated below:

```
# snoop port 23
Using device /dev/hme (promiscuous mode)
myhost.host.com -> remotehost     TELNET C port=33842
   remotehost -> myhost.host.com TELNET R port=33842
myhost.host.com -> remotehost     TELNET C port=33842 \377\372\30\0XTERM\377\360\377\372#\0acsde
   remotehost -> myhost.host.com TELNET R port=33842 \r\n\r\nThis is a secure
myhost.host.com -> remotehost     TELNET C port=33842
   remotehost -> myhost.host.com TELNET R port=33842 \377\373\1\377\375\1If information
myhost.host.com -> remotehost     TELNET C port=33842
   remotehost -> myhost.host.com TELNET R port=33842
myhost.host.com -> remotehost     TELNET C port=33842
myhost.host.com -> remotehost     TELNET C port=33842 m
   remotehost -> myhost.host.com TELNET R port=33842 m
myhost.host.com -> remotehost     TELNET C port=33842
myhost.host.com -> remotehost     TELNET C port=33842 y
   remotehost -> myhost.host.com TELNET R port=33842 y
myhost.host.com -> remotehost     TELNET C port=33842
myhost.host.com -> remotehost     TELNET C port=33842 n
   remotehost -> myhost.host.com TELNET R port=33842 n
myhost.host.com -> remotehost     TELNET C port=33842
myhost.host.com -> remotehost     TELNET C port=33842 a
   remotehost -> myhost.host.com TELNET R port=33842 a
myhost.host.com -> remotehost     TELNET C port=33842
myhost.host.com -> remotehost     TELNET C port=33842 m
   remotehost -> myhost.host.com TELNET R port=33842 m
myhost.host.com -> remotehost     TELNET C port=33842
myhost.host.com -> remotehost     TELNET C port=33842 e
   remotehost -> myhost.host.com TELNET R port=33842 e
myhost.host.com -> remotehost     TELNET C port=33842
myhost.host.com -> remotehost     TELNET C port=33842
   remotehost -> myhost.host.com TELNET R port=33842 Password:
myhost.host.com -> remotehost     TELNET C port=33842
myhost.host.com -> remotehost     TELNET C port=33842 m
   remotehost -> myhost.host.com TELNET R port=33842
myhost.host.com -> remotehost     TELNET C port=33842 y
   remotehost -> myhost.host.com TELNET R port=33842
myhost.host.com -> remotehost     TELNET C port=33842 p
   remotehost -> myhost.host.com TELNET R port=33842
myhost.host.com -> remotehost     TELNET C port=33842 a
   remotehost -> myhost.host.com TELNET R port=33842
myhost.host.com -> remotehost     TELNET C port=33842 s
   remotehost -> myhost.host.com TELNET R port=33842
myhost.host.com -> remotehost     TELNET C port=33842 s
   remotehost -> myhost.host.com TELNET R port=33842
myhost.host.com -> remotehost     TELNET C port=33842 d
   remotehost -> myhost.host.com TELNET R port=33842
myhost.host.com -> remotehost     TELNET C port=33842
   remotehost -> myhost.host.com TELNET R port=33842
myhost.host.com -> remotehost     TELNET C port=33842
   remotehost -> myhost.host.com TELNET R port=33842 Last login: Sat Jun
myhost.host.com -> remotehost     TELNET C port=33842
   remotehost -> myhost.host.com TELNET R port=33842 Sun Microsystems Inc
myhost.host.com -> remotehost     TELNET C port=33842 l
   remotehost -> myhost.host.com TELNET R port=33842 l
myhost.host.com -> remotehost     TELNET C port=33842 s
   remotehost -> myhost.host.com TELNET R port=33842 s
myhost.host.com -> remotehost     TELNET C port=33842
   remotehost -> myhost.host.com TELNET R port=33842 information
myhost.host.com -> remotehost     TELNET C port=33842 e
   remotehost -> myhost.host.com TELNET R port=33842 e
myhost.host.com -> remotehost     TELNET C port=33842
myhost.host.com -> remotehost     TELNET C port=33842 x
   remotehost -> myhost.host.com TELNET R port=33842 x
myhost.host.com -> remotehost     TELNET C port=33842
myhost.host.com -> remotehost     TELNET C port=33842 i
   remotehost -> myhost.host.com TELNET R port=33842 i
myhost.host.com -> remotehost     TELNET C port=33842
myhost.host.com -> remotehost     TELNET C port=33842 t
```

Login Info: **myname**

Password Info: **mypass**

Execute Command: **ls** command typed. Data displayed "**Information**"

Logout Info: **exit** command typed

If **Secure Shell** (SSH) is used to connect to the remote machine, the hacker will have a hard time in decoding the data transmission as shown below. Only the TCP flags like

Acknowledge (ACK), Synchronize (SYN) Sequence numbers, and Close connection (FIN) are in plain text as data is transmitted across the network.

```
#  snoop port 22
Using device /dev/hme (promiscuous mode)
myhost.host.com -> remotehost     TCP D=22 S=33870 Syn Seq=4274531017 Len=0
        Win=32850 Options=<nop,wscale 1,nop,nop,tstamp 164646920 0,nop,nop,sackOK,mss 1460>
remotehost -> myhost.host.com TCP D=33870 S=22 Syn Ack=4274531018 Seq=2904668429
        Len=0 Win=24616 Options=<nop,nop,tstamp 676524644 164646920,nop,wscale
0,nop,nop,sackOK,mss 1460>
myhost.host.com -> remotehost     TCP D=22 S=33870     Ack=2904668430
        Seq=4274531018 Len=0 Win=33304 Options=<nop,nop,tstamp 164646920 676524644>
 remotehost -> myhost.host.com TCP D=33870 S=22     Ack=4274531018
        Seq=2904668430 Len=25 Win=24616 Options=<nop,nop,tstamp 676524803 164646920>
myhost.host.com -> remotehost     TCP D=22 S=33870     Ack=2904672383
        Seq=4274533666 Len=0 Win=33304 Options=<nop,nop,tstamp 164649428 676527152>
 remotehost -> myhost.host.com TCP D=33870 S=22     Ack=4274533666
        Seq=2904672383 Len=96 Win=24616 Options=<nop,nop,tstamp 676527152 164649428>
myhost.host.com -> remotehost     TCP D=22 S=33870     Ack=2904672479
        Seq=4274533666 Len=32 Win=33304 Options=<nop,nop,tstamp 164649429 676527152>
myhost.host.com -> remotehost     TCP D=22 S=33870 Fin Ack=2904672479
        Seq=4274533698 Len=0 Win=33304 Options=<nop,nop,tstamp 164649429 676527152>
 remotehost -> myhost.host.com TCP D=33870 S=22     Ack=4274533699
        Seq=2904672479 Len=0 Win=24616 Options=<nop,nop,tstamp 676527152 164649429>
 remotehost -> myhost.host.com TCP D=33870 S=22 Fin Ack=4274533699
        Seq=2904672479 Len=0 Win=24616 Options=<nop,nop,tstamp 676527153 164649429>
```

Authentication and Encryption Methods

OpenSSH uses the 3DES and Blowfish algorithms[4] for data encryption and decryption. This starts before the authentication begins. For authentication, it uses RSA public key cryptography, One-Time Passwords, and Kerberos for connections. The way RSA authentication works is this: the client maintains the private key, and uploads the public key to the server[5]. The user must supply a password to decrypt the private key. The key is generated using the ssh-keygen command. OpenSSH supports two types of connection protocols. The protocol 1 (version 1.3 and 1.5), the private key is stored in $HOME/.ssh/identity, and the public key is placed into $HOME/.ssh/identity.pub. The content of $HOME/.ssh/identity.pub is then copied to the $HOME/.ssh/authorized_keys file on the remote machine. The protocol 2 (version 2.0) is more secure than protocol 1; the private key is stored in $HOME/.ssh/id_dsa, and the public key is placed into $HOME/.ssh/id_dsa.pub. The contents of $HOME/.ssh/id_dsa.pub should then be copied to the $HOME/.ssh/authorized_keys file on the remote machine. When the user establishes the type of protocol, the users needs to decrypt the contents of $HOME/.ssh/identity to identify themselves in order to gain access to the system. After the connection and authentication process complete, it uses X11 forwarding for encrypting the X windows traffic, and uses port forwarding for encrypting channels for legacy protocols. Encrypting the X windows traffic helps prevent people from sniffing the xterms or inserting malicious commands. Furthermore, it compresses the data before encryption, which can help the performance of a slower network link.

Installation of OpenSSH with TCP Wrappers

Before implementation, download four pieces of software from http://www.sunfreeware.com, and Solaris 8 patch 112438-01 from http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/patch-access. These four items are zlib, openssl, openssh, and tcp_wrappers[6], and patch 112438-01. This patch is required for random number generation by OpenSSH program, which was not pre-loaded in the Solaris 8 operating system. To obtain this patch, type "112438" and on the SUN patch web sites click the "Find Patch" button. It will locate the latest patch release for that patch number. Reboot the system after patch installation in order for it to take effect.

If you do not trust the pre-compiled version, you need to download the source code for openssh, openssl, tcp_wrappers, gcc, lib, make, perl, and prngd (for random number generator daemon). Then, compile each item with "make" utility. For this article, the pre-compiled versions used to demonstrate the OpenSSH setup procedure. This setup procedure[7] can be found at http://www.sunfreeware.com/openssh8.html.

To demonstrate the installation of OpenSSH, I have broken the procedure into five steps. All five scripts can be combined into a single setup script for installation onto multiple hosts. However, one must first install the patch 112438 and reboot the system. This will create the device. Unless the system is rebooted, the last three steps scripts cannot be completed.

After downloading from the Sunfreeware and Sunsolve sites, place the packages, patch, and setup script(s) in /var/tmp directory. Then, unzip the patch by typing "*unzip /var/tmp/112438-01*" and install the patch by typing "*patchadd /var/tmp/112438-01*". After successful patch installation reboot your system before following the setup scripts. In /var/tmp you should have following four files:

zlib-1.2.1-sol8-sparc-local.gz
tcp_wrappers-7.6-sol8-sparc-local.gz
openssl-0.9.7d-sol8-sparc-local.gz
openssh-3.9p1-sol8-sparc-local.gz

Now, you are ready to begin step one script. Uncompress using "gzip", and install with "pkgadd" utilities.

Step One: Unzip and Install Packages Script

```
gzip    -d zlib-1.2.1-sol8-sparc-local.gz
gzip    -d tcp_wrappers-7.6-sol8-sparc-local.gz
gzip    -d openssl-0.9.7d-sol8-sparc-local.gz
gzip    -d openssh-3.9p1-sol8-sparc-local.gz
pkgadd  -d zlib-1.2.1-sol8-sparc-local
pkgadd  -d tcp_wrappers-7.6-sol8-sparc-local
pkgadd  -d openssl-0.9.7d-sol8-sparc-local
pkgadd  -d openssh-3.9p1-sol8-sparc-local
```

After installation, the following packages are added to your system. To verify the package, version, date, directory installed in, you can type: "pkginfo -l SMCzlib", "pkginfo -l SMCtcpw", and so forth.

SMCzlib	*zlib*	*(sparc) 1.2.1*
SMCtcpw	*tcp_wrappers*	*(sparc) 7.6*
SMCssl	*openssl*	*(sparc) 0.9.7d*
SMCossh3	*openssh*	*(sparc) 3.9p1*

On step two, add a user and group account "sshd", and create the /var/empty directory for the "sshd" user. The "sshd" account must be locked, and the directory must not contain any data. This is a new security feature called "Privilege Separation[8]" incorporated in the OpenSSH version 3.4 released in July 2002. Privilege separation prevents root compromise if the sshd process becomes corrupted. The sshd daemon forks two processes: one parent and one child. The parent process, which runs as root, monitors the unprivileged child process. The child process, which runs as unprivileged "sshd" user, processes the network data. Further explanation on Privilege Separation and usage can be found at http://www.sunfreeware.com/README.privsep.

Step Two: Setup the sshd user and /var/empty director script

```
echo " Creating sshd user..."
mkdir /var/empty
chown root:sys /var/empty
chmod 755 /var/empty
groupadd sshd
useradd -g sshd -c 'sshd privsep' -d /var/empty -s /bin/false sshd

echo " Please modify the /etc/hosts.allow file to look like the following example "
echo " sshd: remote.host.IP.#/remote.host.Netmask: ALLOW "
```

After creating the "sshd" account, limit network access to the system by creating and configuring the hosts.allow and host.deny files. Without these files, the system is wide open to crackers everywhere. To configure the host.allow file, you add each individual or a small range of IP addresses, and list individual services in the hosts.allow file. Limit the usage of "ALL" like "ALL:ALL:ALLOW", which permits any host to establish connection to any port that is listening on the host. If a wide range of hosts to access the system, try to identify the subnet IP address(es). Grouping them to a particular subnet, narrows the IP ranges. Then, specify a syntax like "Services:remote.host.netid./remote.Sub.net.mask:ALLOW". For an example, "sshd:192.168.10./255.255.255.0:ALLOW". This means the IP addresses in the same subnet will be able to connect to the system for the specified service. It is even better, to specify the single IP addresses like "sshd: remote.host.netid.IP#/remote.Sub.net.mask: ALLOW:". By adding individual IP addresses will lower the chance of someone breaking into the machine. Moreover, block any unwanted traffic to the system by adding "ALL: ALL: DENY" in the hosts.deny file.

/etc/hosts.allow file

```
# Specifying Individual IP Address
sshd:  remote.host.IP: ALLOW
in.ftpd: remote.host.IP: ALLOW
in.telnetd: remote.host.IP: ALLOW
# Specifying a  small range of IP Address - have same prefix
sshd:  remote.host.netid./remote.sub.net.mask:Allow
in.ftpd: remote.host.netid./remote.sub.net.mask:Allow
in.telnetd: remote.host.netid./remote.sub.net.mask:Allow
```

/etc/hosts.deny file
ALL: ALL: DENY

Once you have created the hosts.allow and hosts.deny files and blocked all unknown IP addresses into your system, the log will contain messages like " *Jul 28 04:47:14 hostname in.ftpd[28734]: [ID 947420 mail.warning] refused connect from remote.host.IPaddress".*

Important Note: Reboot the system, before the step three script. Otherwise, you will encounter the "PRNG not seeded" error message. You can attempt the random script mentioned at http://halplant.com:88/software/Solaris/scripts/setup_random. However, if following error messages appear after running this setup script, you must reboot the system in order to fix the devices /dev/random and /dev/urandom.

Jul 1 17:25:54 hostname genunix: [ID 806237 kern.warning] WARNING: ***ddi_installdrv: no major number for random***
Jul 1 17:25:54 hostname genunix: [ID 703013 kern.warning] WARNING: ***mod_installdrv: Cannot install random***

In the step three scripts, OpenSSH uses ssh-keygen to generate, manage, and convert authentication keys[9]. The ssh-keygen creates RSA public and private key pairs for SSH protocol version 1, and RSA or DSA public and private key pair for SSH protocol version 2. To setup SSH, use ssh-keygen to generate the public and private key. Specify the "-t" option for either RSA or DSA key, the "-f" option for a file location to write the private key and a passphrase to encrypt the private key. Specify a passphrase with the -N "*My Passphrase Sentences*" option. If the passphrase is forgotten, the public key must be re-generated and re-copied to the remote machines. A lost passphrase is not recoverable. Specifying the passphrase is the best security tightening method. However, it may create additional administration duties. Therefore, if there are many systems and there is not a concern about the additional security feature, leave the passphrase blank. This means that when a user first establishes a SSH session, the user will be prompted to add the authentication key to his/her machine. It creates this key and places it in $HOME/.ssh/identity, $HOME/.ssh/id_dsa or $HOME/.ssh/id_rsa.

Step Three: Generate public and private rsal, dsa, rsa keys script

Generate without a passhrase

/usr/local/bin/ssh-keygen -t rsa1 -f /usr/local/etc/ssh_host_key -N ""
/usr/local/bin/ssh-keygen -t dsa -f /usr/local/etc/ssh_host_dsa_key -N ""
/usr/local/bin/ssh-keygen -t rsa -f /usr/local/etc/ssh_host_rsa_key -N ""

or

Generate with a passhrase

/usr/local/bin/ssh-keygen -t rsa1 -f /usr/local/etc/ssh_host_key -N "my phassphrase"
/usr/local/bin/ssh-keygen -t dsa -f /usr/local/etc/ssh_host_dsa_key -N " my phassphrase"
/usr/local/bin/ssh-keygen -t rsa -f /usr/local/etc/ssh_host_rsa_key -N " my phassphrase"

Running this script the rsal, dsa, and rsa key are generated as displayed below.

*Generating public/private **rsa1 key** pair.*
Your identification has been saved in /usr/local/etc/ssh_host_key.
Your public key has been saved in /usr/local/etc/ssh_host_key.pub.
The key fingerprint is:
b2:39:13:c2:91:23:4f:ee:79:3d:02:d8:d6:2f:1e:08 root@myhost
*Generating public/private **dsa key** pair.*
Your identification has been saved in /usr/local/etc/ssh_host_dsa_key.
Your public key has been saved in /usr/local/etc/ssh_host_dsa_key.pub.
The key fingerprint is:
77:6a:fa:57:aa:65:21:30:da:a4:ce:7a:24:33:8a:39 root@myhost
*Generating public/private **rsa key** pair.*
Your identification has been saved in /usr/local/etc/ssh_host_rsa_key.
Your public key has been saved in /usr/local/etc/ssh_host_rsa_key.pub.
The key fingerprint is:
4d:32:9e:54:b2:2f:13:f4:ea:66:bf:19:f9:ea:a6:ec root@myhost

Step four, first backs up the /etc/inet/services and /etc/inet/inetd.conf files. Then, it adds the SSH services to the services file, adds the SSH wrapper with tcp_wrappers in the inetd.conf file.

Step Four: Add SSH services port to services file, configure inetd.conf file, and create sshd start script

```
DATE=`date '+%m/%d'`
DAY=`date '+%d'`
HOUR=`date '+%H'`
MONTH=`date '+%m'`
YEAR=`date '+%y'`
MIN=`date '+%M'`
echo "Backup and add following line to services"
cp -p /etc/inet/services      /etc/inet/services.$MONTH$DAY$YEAR
echo "secureSSH              22/tcp    # Secure Shell port                    " >> /etc/services
echo "Backup and add following line to the inetd.conf"
cp -p /etc/inet/inetd.conf    /etc/inet/inetd.conf.$MONTH$DAY$YEAR
echo "secureSSH     stream  tcp     nowait  root    /usr/local/sbin/tcpd   /usr/local/sbin/sshd -i
                                                                          " >> /etc/inet/inetd.conf
```

After you running the step four script, tcp_wrappers will help monitor and filter incoming SSH request[10]. The wrapper program logs the name of the client host and the requested services. It uses the RFC 931 and other protocols to identify the owner of the connection host. This protects against hosts that try to spoof another host. This can also be useful when looking for signs of intrusion. To set up the SSH listening services, add SSH service in the /etc/inet/services file. Illustrated below, the SSH services are named "secureSSH" and placed in the /etc/inet/services file. This name must also be the same in the /etc/inet/inetd.conf file.

/etc/inet/services file

```
secureSSH                22/tcp    # Secure Shell port
```

/etc/inet/inetd.conf file

```
secureSSH     stream  tcp     nowait  root    /usr/local/sbin/tcpd   /usr/local/sbin/sshd -i
```

The step five script restarts inetd. The script obtains the inetd process, and issues the command "kill -HUP process-id". Restarting the inetd deamon will enable the SSH listener port.

Step Five: Restart inetd (final step)

```
#!/bin/sh pid=`/usr/bin/ps -e | /usr/bin/grep inetd | /usr/bin/sed  -e 's/  ^ *//'  -e 's/   .*//'`
        if [ "${pid}" != "" ]
        then
        /usr/bin/kill -HUP ${pid}
        fi
```

Identify that ssh starts on port 22 by typing "netstat -a |grep ssh". In this case, I gave my service the name "secureSSH". I will need to type " netstat -a |grep secureSSH" shown as follow:

```
netstat -a |grep secureSSH
*.secureSSH          *.*              0      0 65536      0 LISTEN
myhost.secureSSH  remotehostIP1.1926   9324     0 65268    20 ESTABLISHED
myhost.secureSSH  remotehostIP2.2055   8392     0 65268     0 ESTABLISHED
myhost.secureSSH remotehostIP3..3324   8328     0 65268     0 ESTABLISHED
```

As you can see, three SSH connections are established, and one listening port named "*.secureSSH". Once the connection is established, two processes are started for you.

```
myhost#  ps -ef |grep sshd
dog   929   926  0   Jul 15 ?       0:00 sshd -i
root   926   203  0   Jul 15 ?       0:01 sshd -i
cat   1523  1520  0 10:28:15 ?       0:07 sshd -i
root   1520   203  0 10:28:01 ?       0:01 sshd -i
```

This new feature of "Privilege Separation" in OpenSSH version 3.4 is to prevent the corruption of the unprivileged child process that would compromise the system. The privileged parent process "id: 1520" monitors the unprivileged child process "id 1523". If child authentication is unsuccessful, it will not grant access unless the parent process has reached the same conclusion.

At this point, you have successfully configured OpenSSH on your system. To verify the SSH version installed on the system type "ssh -V". You will see the information like "OpenSSH_3.9pl, SSH protocols 1.5/2.0, OpenSSL 0x0090604f".

Now, let us examine the messages and syslog. As a user connects to the system via SSH you will see a /var/log/syslog message like "*Jun 1 16:50:36 remotehost sshd[24887]: [ID 927837 mail.info] connect from myhost.host.com*". To improve or isolate message log capture, add the "auth.info" line in /etc/syslog.conf, and create a "authlog" file in /var/log directory. Create by typing "*cat /dev/null > /var/log/authlog*". This creates an empty file called "authlog". Restart syslog for this configuration to take effect. To restart the syslog deamon, type "*kill -HUP process-id*", or manually stop and start with /etc/rc2.d/S74syslog.

/etc/syslog.conf file

```
auth.info                                /var/log/authlog
```

```
myhost#  ps -ef | grep syslogd
```

*root **348** 1 0 May 20 ? 0:00 /usr/sbin/syslogd*
*myhost# **kill -HUP 348***

With the auth.info configured, both successful and unsuccessful logins will be logged in the /var/log/authlog file as follows:

/var/log/authlog_file
Jun 1 17:08:19 myhost sshd[25048]: [ID 800047 auth.info] Accepted password for myname from 192.9.99.80 port 414

To trace incoming inetd connections, edit /etc/init.d/inetsvc and add "-t" flag as shown below.

/usr/sbin/inetd -s -t

After successfully installing the OpenSSH and testing the SSH connection, disable telnet, ftp, rsh, login, and shell services. By commenting out or removing telnet, shell, and ftp from the /etc/inet/inetd.conf file. This forces users to access the forces users to access the SSH services. In addition, disable any services you do not need to run or use the tcp_wrappers to protect the services that you need to run. Disabling unneeded services will prevent your system being subject to denial of service attacks. After changing the /etc/inet/inetd.conf file, restart the inetd deamon in order for it to take effect. Once this is done, a "connection refused" message is logged as someone uses the unavailable utilities.

*# **telnet "remotehost"***
Trying remotehostIP..
telnet: Unable to connect to remote host: Connection refuse

Another Note: If you need X11 and TCP forwarding capabilities to encrypted channel, must enable the sshd_config and ssh_config files. It is disabled in the default installation. The reason was due the malicious user can gain unauthorized connections to the system and was reported on CERT Vulnerability Note VU#363181[11]. Use it at your own risk! This is done by uncommenting and changing from "X11Forwarding no" to "X11Forwarding yes" in the sshd_config file, and changing from "ForwardAgent no" and "ForwardX11 no" to "ForwardAgent yes" and "ForwardAgent yes" in the ssh_config file. Or, use "-X" option to enable X-Term windows automatically displayed on your local machine. You should not manually set DISPLAY.

Using SSH on UNIX

OpenSSH program comes with several commands. The functionality of the commands is as follow:

ssh[12] - This command logs into remote machine, executes commands on a remote machine. It replaces the telnet, rlogin, and rsh. The option most commonly uses are as follow:

-l userid	- log in with this user identity.
-V	- display the OpenSSH, OpenSSL, and protocols version.
-1	- force protocol version 1.
-2	- force protocol version 2.

Type "ssh" will list the options and usage information.

sshd[13] - This deamon listens for incoming connections, and provides encrypted communications between two hosts. It starts on the server side only.

scp[14] - This command copies files from one machine to another.

ssh-keygen[9] - This generate RSA or DSA authentication key.

ssh-agent[15] - It is a authentication agent. It unencrypts the RSA private key when communicate with the remote machine.

ssh-add[16] - This adds the private keys to the files $HOME/.ssh/id_rsa, $HOME/.ssh/id_dsa and $HOME/.ssh/identity.

sftp-server[17] - This talk to the server side of SFTP protocol.

sftp[18] - This command securely transfers file from one machine to another. It replaces the ftp command.

ssh-keycscan[19] - This utility build and verify from the ssh_known_hosts file when the users accessing the machine..

To use "ssh" on a UNIX machine, type "ssh remotehost -l userid" at the command prompt. You will be prompted to add the authentication key. After answering "yes" to accept the RSA key fingerprint, you will be prompted to enter your password. If you answer "yes" to accept the RSA key and "Are you sure you want to continue conecting (yes/no)?" question, you will not be prompted again when you re-connect to the system.

#./ssh "remotehost" -l "userid"
The authenticity of host 'remotehost (remotehostIP)' can't be established
RSA key fingerprint is 82:bc:70:ee:1e:e4:36:c4:fb:41:0b:21:ec:de:95:83
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'remotehost, remotehostIP' (RSA) to the list of known hosts
root@ remotehost 's password:
Last login: Sat Jun 1 17:26:29 2002 from remotehost@host.com
Sun Microsystems Inc. SunOS 5.8 Generic Patch October 2002
#

The use file transfer utility "scp" replaces "rcp" is as follow:

To copy file from remote host, type:
"scp userid@remotehost:/remote/filename /local/directory/copyInto".

To copy file to remote host, type:
"scp filename userid@remotehost:/remote/directory/copyInto".

System Troubleshooting:

An error message like "**ssh_exchange_identification: Connection closed by remote host**", means your remote system does not permit your connection. Check the remote system to see if sshd and ip address are enabled in the /etc/hosts.allow file. An sample /etc/hosts.allow file configuration is shown below:

/etc/hosts.allow file
...
sshd: remotehostIP/remotehostNetmask: ALLOW
sshd: remotehostIP: ALLOW
...

If you have problem using scp or ssp, you can diagnose the failing connection by adding the -v option.

Using and installing SSH client on Microsoft Window

TeraTerm is a free windows emulator terminal for Microsoft Windows[20]. This program enables the user to establish sessions like telnet, ssh, serial port connection, and so on. You can download TeraTerm at http://hp.vector.co.jp/authors/authors/VA0222416/ttermp23.zip. After the download is complete, double click "ttermp23.zip". It will force the file to be unziped into the target directory c:\temp\ttermp23. Double click "setup.exe" in c:\temp\ttermp and follow the installation instructions. It will install into the default directory c:\Program Files\Ttermpro. Next, install the "ttssh[21]" addon program. Obtain this by downloading it from http://www.zip.com.au/~roca/ttssh.html. This will allow you connect via SSH. First, unzip the "ttsshencode154.zip" into c:\temp\ttermssh. Copy the ttxssh.dll, libeay32.dll, and ttssh.exe to the TeraTerm program directory c:\program Files\Ttermpro. For easy access, copy "ttssh.exe" and paste the shortcut onto you desktop. Now, establish a connection to the system by double clicking "ttssh.exe", or by double clicking TeraTerm shortcut icon you created earlier. Once a TeraTerm session is established, enter a remote "Host name" and the port number. The port number is optional as it is automatically set when the SSH service button is clicked. Click the "OK" button to connect as shown on *figure 1*.

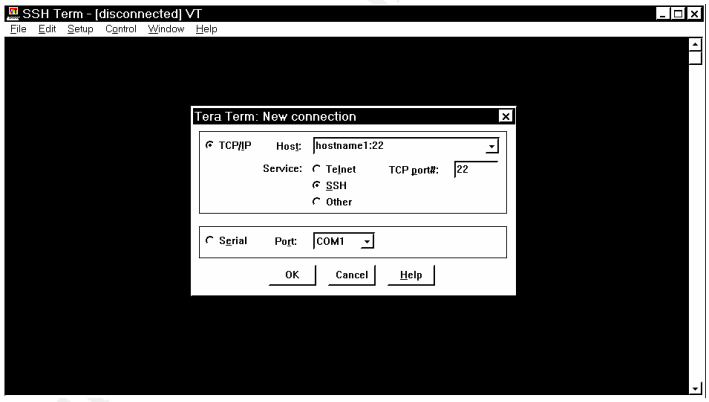

Figure 1: Establish new Connection using Tera Term

Next, select one of the authentication methods:

> **Password only**: Enter your password in the "passhrase" box. It will encrypt your password and transmit to the server.
> **RSA**: Select the file containing your RSA private key. This file can be generated by ssh-keygen.
> **rhosts and rhosts with RSA: E**nter you user id in the "user name" box and the passphrase in the "passphrase" box.
> **TIS Challenge/Response:** You do not need to enter a password. The server will send a challenge message and you need to response to the message.

If you choose the "Password only" radio button, it will first check the "ssh_known_hosts" file to see if your machine knows about the server you are trying to contact. This is to

protect you from hostile machines pretending to be your server[22]. If your machine tries to contact the server that does not have itself in the known hosts file, it will display warning messages and prompt you to add the host authentication key on to your PC. If you know that the server is legitimate, check "Add this new key to the known hosts list", and click continue as shown on *figure 2*. If you do not answer the question, then next time you connect, that host will prompt you again.

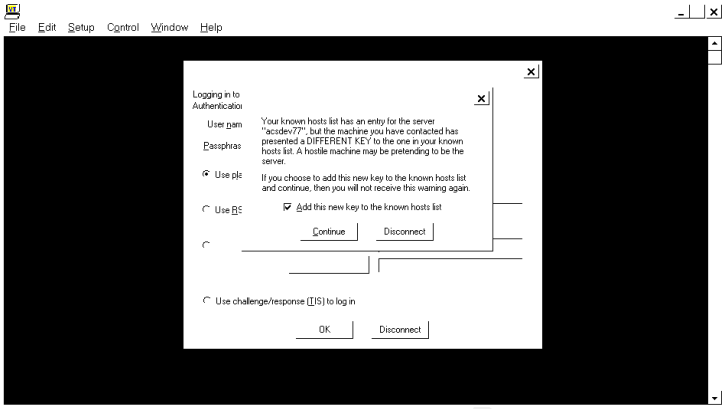

Figure 2: Authentication key on your system

After adding the authentication key to your Windows machine, you will be prompted to enter your login id in "User Name" box, and password in the "Passphrase" box as shown on *figure 3*.

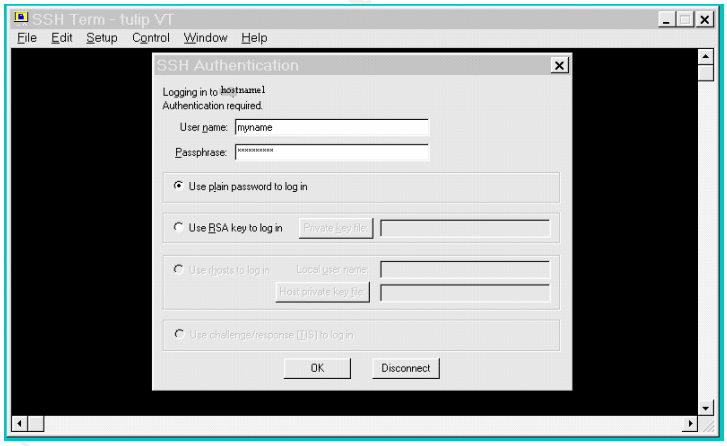

Figure 3: Enter User Name and Password.

If your server accepts your authentication, your shell session will start. You are successfully connected your system using SSH program from your PC. You can also execute the program on the DOS line, by typing "*ttssh hostname:22 /ssh*" to start an SSH connection.

Alternatively, if you select RSA as your authentication mechanism, you first need to create the host key in your local machine by using ssh-keygen. Then, copy the identity.pub to the system you need to connect by adding into the authorized_key file. Once, you have done this, the system will allow your machine to access into the system.

The TeraTerm program is an elegant terminal emulator program. However, it does not come with the ssh-keygen. You need to download and install OpenSSH version 3.4 for windows from http://www.networksimplicity.com/openssh. Follow the setup instructions and select the client version. Once installed, execute ssh-keygen2.exe[23] from the c:\Program Files\SSH Communications Security\SSH Secure Shell folder to generate the host key.

The strongest authentication and best security tightening method is to use the RSA authentication mechanism. This means the server will reject any machines that are not on the authorized_key file. However, this will create additional administrative duties if you have many servers.

Conclusion

Telnet, rlogin, ftp are insecure and vulnerable to exploitation by hacker, malicious users, and attackers. Implementing SSH will prevent or discouraging such people from breaking into the system. To further tighten operating system security, regularly update the system operating patches, disable ip forwarding, multicasting, and any services you do not need to start up. Frequently check log files for any unusual activities. Further, protect your system by implementing software like Tripwire, Snort, and SunScreen. Finally, subscribe to CERT or check CIAC site for new vulnerabilities and security issues. See *Appendix A* on how to subscribe to CERT.

Appendix A

To subscribe to the CERT Advisory Mailing List, e-mail to majordomo@cert.org. In the body of the message, type "subscribe cert-advisory". You can also obtain how to subscribe information at URL: http://www.cert.org/contact_cert/certmaillist.html

Software Links:

Main Sunfreeware.com site
URL: http://www.sunfreeware.com

Openssh-3.4pl-sol8-sparc-local.gz
URL: ftp://ftp.sunfreeware.com/pub/freeware/sparc/8/openssh-3.4p1-sol8-sparc-local.gz
zlib-1.1.4-sol8-sparc-local.gz
ftp://ftp.sunfreeware.com/pub/freeware/sparc/8/zlib-1.1.4-sol8-sparc-local.gz
tcp_wrappers-7.6-sol8-sparc-local.gz
ftp://ftp.sunfreeware.com/pub/freeware/sparc/8/tcp_wrappers-7.6-sol8-sparc-local.gz
openssl-0.9.6d-sol8-sparc-local.gz
ftp://ftp.sunfreeware.com/pub/freeware/sparc/8/openssl-0.9.6d-sol8-sparc-local.gz

Teranishi, T. Tera Term Home Page.
URL:http://hp.vector.co.jp/authors/VA002416/teraterm.html
O'Callahan, Robert. TTSSH: An SSH Extension to Teraterm.
URL:http://www.zip.com.au/~roca/ttssh.html
Bradshaw, Mark "OpenSSH on Window v3.4-1" Network Simplicity
URL: http://www.networksimplicity.com/openssh.

List of References:

[1] Myers, Eric. "Please use Secure Shell (SSH) instead of Telnet or rsh/rcp/rlogin." 21 Nov. 2001 URL: http://feynman.physics.lsa.umich.edu/~myers/help/SecureShell.html
[2] Gross, Grant. "OpenSSH: the five-year trademark itch." 14 Feb. 2001
URL: http://www.theregister.co.uk/content/4/16928.html
[3] OpenBSD. "OpenSSH" Project History and Credits. 20 Jan. 2002
URL: http://www.openssh.com/history.html
[4] OpenBSD. "OpenSSH" Feaures. 19 May 2002
URL:http://www.openssh.com/features.html
[5] Mates, Jeremy. "SSH Use and Users" Jun 1999
URL: http://cellworks.washington.edu/sage/1999/06/ssh.pdf
[6] Venema, Wietse. "README 1.30." This is the 7.6 version of the TCP/IP daemon wrapper package." 03 Mar. 1997
URL: http://www.sunfreeware.com/README.tcpwrappers
[7] Christensen, Steven and Associates, Inc. "Installing OpenSSH Packages for SPARC and Intel/Solaris 8." 10 Jul 2002

URL:http://www.sunfreeware.com/openssh8.html
[8] Provos, Niels. "Privileged Separation OpenSSH." 24 Jul 2002
URL:http://www.citi.umich.edu/u/provos/ssh/privsep.html
[9] Galbraith, J. and Thayer, R. "OpenBSD" Manual Pages. SSH-KEYGEN(1)" Berkeley Software Design, Inc. 25 Sep. 1999.
URL: http://www.openbsd.org/cgi-bin/man.cgi?query=ssh-keygen
[10] Schneider, Wolfram. "OpenBSD Manual Pages - SSH(1)" Berkeley Software Design, Inc. 25 Sep. 1999.
URL: http://www.openbsd.org/cgi-bin/man.cgi?query=ssh
[11] Hernan, Shawn and Van Ittersum, Shawn. "Vulnerability Note VU#363181" Carnegie Mellon University. 30 May 2002.
URL: http://www.kb.cert.org/vuls/id/363181
[12] Schneider, Wolfram. "OpenBSD Manual Pages - SSH(1)" Berkeley Software Design, Inc. 25 Sep. 1999. URL: http://www.openbsd.org/cgi-bin/man.cgi?query=ssh
[13] Schneider, Wolfram. "OpenBSD Manual Pages - SSHD(8)" Berkeley Software Design, Inc. 25 Sep. 1999. URL: http://www.openbsd.org/cgi-bin/man.cgi?query=sshd
[14] Schneider, Wolfram. "OpenBSD Manual Pages - SCP(1)" Berkeley Software Design, Inc. 25 Sep. 1999. URL: http://www.openbsd.org/cgi-bin/man.cgi?query=scp
[15] Schneider, Wolfram. "OpenBSD Manual Pages - SSH-AGENT(1)" Berkeley Software Design, Inc. 25 Sep. 1999.
URL: http://www.openbsd.org/cgi-bin/man.cgi?query=ssh-agent
[16] Schneider, Wolfram. "OpenBSD Manual Pages - SSH-ADD(1)" Berkeley Software Design, Inc. 25 Sep. 1999.
URL: http://www.openbsd.org/cgi-bin/man.cgi?query=ssh-add
[17] Friedl, Markus. "OpenBSD Manual Pages - SFTP-SERVER(8)" Berkeley Software Design, Inc. 30 Aug. 2000.
URL: http://www.openbsd.org/cgi-bin/man.cgi?query=sftp-server
[18] Schneider, Wolfram. "OpenBSD Manual Pages - SFTP(1)" Berkeley Software Design, Inc. 4 Feb. 2001. URL: http://www.openbsd.org/cgi-bin/man.cgi?query=sftp
[19] Schneider, Wolfram. "OpenBSD Manual Pages - SSH-KEYSCAN(1)" Berkeley Software Design, Inc. 1 Jan. 1996.
URL: http://www.openbsd.org/cgi-bin/man.cgi?query=ssh-keyscan
[20] Teranishi, T. "Tera Term Home Page." 9 Aug. 1999
URL: http://hp.vector.co.jp/authors/VA002416/teraterm.html
[21] O'Callahan, Robert. "TTSSH: An SSH Extension to Teraterm." Mar 2001
URL:http://www.zip.com.au/~roca/ttssh.html
[22] O'Callahan, Robert. "TTSSH: An SSH Extension to Teraterm version 1.5." Installation Instructions. Mar 2001 URL:http://www.zip.com.au/~roca/ttsshdoc.html
[23] Bradshaw, Mark "OpenSSH on Window v3.4-1" Network Simplicity. Jul 2002
URL: http://www.networksimplicity.com/openssh.

Last Updated: October 21st, 2017

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS San Diego 2017	San Diego, CAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Seattle 2017	Seattle, WAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, AE	Nov 04, 2017 - Nov 16, 2017	Live Event
SANS Milan November 2017	Milan, IT	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Amsterdam 2017	Amsterdam, NL	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Miami 2017	Miami, FLUS	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Paris November 2017	Paris, FR	Nov 13, 2017 - Nov 18, 2017	Live Event
Pen Test Hackfest Summit & Training 2017	Bethesda, MDUS	Nov 13, 2017 - Nov 20, 2017	Live Event
SANS Sydney 2017	Sydney, AU	Nov 13, 2017 - Nov 25, 2017	Live Event
GridEx IV 2017	Online,	Nov 15, 2017 - Nov 16, 2017	Live Event
SANS San Francisco Winter 2017	San Francisco, CAUS	Nov 27, 2017 - Dec 02, 2017	Live Event
SANS London November 2017	London, GB	Nov 27, 2017 - Dec 02, 2017	Live Event
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZUS	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Khobar 2017	Khobar, SA	Dec 02, 2017 - Dec 07, 2017	Live Event
SANS Austin Winter 2017	Austin, TXUS	Dec 04, 2017 - Dec 09, 2017	Live Event
SANS Munich December 2017	Munich, DE	Dec 04, 2017 - Dec 09, 2017	Live Event
European Security Awareness Summit & Training 2017	London, GB	Dec 04, 2017 - Dec 07, 2017	Live Event
SANS Bangalore 2017	Bangalore, IN	Dec 11, 2017 - Dec 16, 2017	Live Event
SANS Frankfurt 2017	Frankfurt, DE	Dec 11, 2017 - Dec 16, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DCUS	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LAUS	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS SEC460: Enterprise Threat Beta	San Diego, CAUS	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Amsterdam January 2018	Amsterdam, NL	Jan 15, 2018 - Jan 20, 2018	Live Event
Northern VA Winter - Reston 2018	Reston, VAUS	Jan 15, 2018 - Jan 20, 2018	Live Event
SEC599: Defeat Advanced Adversaries	San Francisco, CAUS	Jan 15, 2018 - Jan 20, 2018	Live Event
SANS Berlin 2017	OnlineDE	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced