Global Information Assurance Certification Paper

Jeff Miller
Assignment Version 1.7

Solaris 8 Installation for the Purpose of a Secure File Transfer Server

Intent

This document is an instruction set for the installation of a Solaris 8 server. The server is to be used as a point to allow offsite employees, customers, or partners to securely deposit files onto or retrieve files from using any Internet connected server. No shell access will be provided for users. The only functionality users are allowed is the ability to get or post files within their own accounts. The definition of a secure copy for the purposes of this paper is one that uses an encrypted connection and the only authentication allowed to make the connection is public key authentication.

Software Setup Overview

We will setup an NTP client for the purposes of keeping our server synchronized to a reasonable time for logging and auditing. Also, a DNS client will be setup for logging of connections. SSH server software will be installed, as it will provide the main functionality of the server. All other network services are deemed unnecessary and will be disabled.

In addition to the software setup, we will also need to set up a chrooted environment, allowing the user access to only files and commands necessary for the file copying. This will keep the user out of important or sensitive system information.

Network Placement

This document will not address firewall configuration or network design. Given the fact that this system will be Internet connected, a firewall is highly recommended. This is the type of server that would be placed in a DMZ with respect to firewall setup. That is to say it would be protected from the Internet and the intranet.

Hardware Setup, Inventory, and Physical Location

The Sun Enterprise-250 server for this exercise is configured with 2-300 MHz processors, 2-18 GB hard drives, 256 MB of memory, a SCSI CD-ROM, 1-100 Mb Ethernet card, 1-SCSI 4mm tape drive and dual power supplies. It is important to record this information

about the system for future use. We will be using a serial console to manage this system. The network connection will not be plugged in until necessary (at the very end of the procedure). The server will reside be in a locked data center with tightly controlled physical access. This is important because physical access could severely compromise the security of this server.

Open Boot Prom Settings

The first time you turn on the server you should get the "ok" prompt. If the server begins to boot just send a break signal to the console. Using a standard telnet command in Unix this can be accomplished by escaping to the telnet command prompt with **CTRL-]** then type **send break**. You should now see the "ok" prompt.

Verify **auto-boot?** is set to true using **setenv auto-boot? true**. In case of a power failure, the system will attempt to come back up. Otherwise the system will wait at the "ok" prompt upon any reset.

Check out the banner from when the system was first booted. Record this information and keep it with other system configuration information. Then change the banner to only include a message to tell unauthorized users they are not welcome. You can do this with the **setenv oem-banner "Authorized access only."** and **setenv oem-banner? true**. This will also hide the standard oem-banner if you are concerned about giving away that information.

Installing Solaris 8

Here are the basic steps to go through for the initial install of Solaris 8 for this server.

1. Insert the Solaris 8-disk 1 CD-ROM.
2. From the "ok" prompt, type **boot cdrom** to start the install.
3. Select Language, Locale, and terminal type.
4. When prompted to setup networking check no.
5. Set your hostname, time zone, date and time information.
6. Choose the "Initial" install from the "Solaris Interactive Installation" menu.
7. Do not select any geographic regions.
8. Select 64-bit operating system support.
9. Select the "Core System Support" software bundle.
10. Select your boot disk (c0t0d0 in our case). Ensure it is setup as a boot disk.
11. Do not preserve any previous data.
12. Select automatic layout for the file systems. You should layout /, /opt, /usr, /var, and swap.
13. Now customize the sizes for the file system layout. Use the following as a guideline.
    - Setup / to 301 MB.
    - Setup /var to 4001 MB.

- Setup swap to 257 MB (or about the size of memory).
- Setup partition 4 with no name but 5 MB of space for later installation of the Solaris Disk Suite tool.
- Setup /opt to 4001 MB.
- Setup /usr to 1000 MB.
- Rename /export/home to /home and give it 7691 MB. This is where our users files will be.

14. At this point we can't mount any remote file systems. So skip that option.
15. Check reboot after installation.

Root Password and Root Access

Login as root and set the root password using the **passwd** command. Pick a password that will not be easily guessed but you can remember easily. Alternatively, root passwords can be stored in a highly secured box, on the premises; in case of an administrator has a mishap and can't recall the password.

Also, you don't want root logging in from just anywhere. Edit **/etc/default/login** make sure the **CONSOLE** variable is at least set to **/dev/console** to allow root logins on the console device or set it to **/dev/null** to allow no root logins (only su).

Install Additional Software Packages

We need a few things above and beyond the core operating system. Here we will install network time protocol and the accounting packages. To do this, mount the Solaris Software Packages Disk 1. Change directories into the Product directory and run **pkgadd –d . SUNWnptr . SUNWntpu**. Answer 'y' to any the questions. Now unmount that CD-ROM and mount Solaris Software Packages Disk 2. Change directories to the Product directory and execute **pkgadd –d . SUNWaccr SUNWaccu**. We will configure these packages later in this document. For reference, mount and unmount the CD-ROM as follows (assume that the c0t6d0 is the CD-ROM device file):

- **mount –F hsfs –o ro /dev/dsk/c0t6d0s0 /mnt**
- **cd /mnt/Solaris_8/Product**
- Run the pkgadd.
- **cd /**
- **umount /mnt**

Install Operating System Patches

It is very important to keep up to the latest patch set. Security fixes are coming out all the time. You can do this on the server with the following instructions:

1. On a system that has network access, download the latest recommended patch set and its associated README file from **ftp://sunsolve.sun.com/pub/patches/**. You should download the **8_Recommended.README**, **8_Recommended.zip**, and **CHECKSUMS** files. The zip file contains the actual patches and installation scripts. The README file should contain all the information about the patches you will be installing and how to install them. You should read the README file closely. The CHECKSUMS file contains the md5 checksum for all downloads in the patches directory.
2. Verify the patch set by running **md5sum 8_Recommended.zip**. Check the output of this command against the information listed in the CHECKSUMS file. If it does not match do not continue! It could mean that your download has been compromised, or at least not complete or what you expect.
3. Once have verified the patch set, unzip it using the **unzip 8_Recommended.zip** command.
4. Now copy the patch directory to tape for export to the new system. You can do this using **tar cvf /dev/rmt/0 *patchdir***. Here rmt0 is assumed to be your tape device and patchdir is the name of the directory you unzipped 8_Recommended.zip into.
5. On the system we're installing to, copy the patches from the tape to a persistent temporary directory (not /tmp). For example, **mkdir –m 700 /opt/patchtmp ; cd /opt/patchtmp ; tar xvf /dev/rmt/0**.
6. Change directories into the patch directory.
7. Run **./install_cluster** command inside that directory. Watch for errors. You will probably see some errors from patches for packages that are not installed.
8. Reboot after the patch install is finished with the **reboot** command.
9. Check the patch error log when the system comes back up. It is located in **/var/sadm/install_data/Solaris_8_Recommended_log**.

Remove Unnecessary Startup Scripts

We want to remove any startup scripts we do not need. This includes network services cache daemon, any scripts attempting auto configuration, rpc, nfs, sendmail (we will run sendmail from cron later), cacheos, dhcp, and PRESERVE.

I did this in one fell swoop using the following commands:

- **cd /etc; rm `for file in cacheos dhcp sysid.net sysid.sys autoinstall rpc nscd nfs autofs cachefs ldap PRESERVE sendmail ; do ; echo "[ir]*[t0123S].d/*${file}* " ; done`**

This will delete the startup scripts in **/etc/init.d** and also in all the associated links in the **/etc/rc** directories. You want to get rid of these so they cannot be accidentally re-linked or re-run at a later date. Also, with out the scripts there is no confusion about what should be running.

Modifications to Existing Startup Scripts

There are still a few startup scripts that contain items we don't want running. Create a new version of **/etc/init.d/inetsvc**. We will call it **/etc/init.d/inetsvc.modified**. It should contain only the following.

```
#!/sbin/sh
#
/usr/sbin/ifconfig –au netmask + broadcast +
```

This gets rid of many things including multicasting, DNS servers (if configured), and inetd. Getting rid of inetd takes care of popular services like telnet, ftp, rsh, remsh, etc. Now we need to remove all the links to the original inetsvc script. Execute the following:

- **rm /etc/rc[012S].d/[KS]*inetsvc; ln –s /etc/init.d/inetsvc /etc/rc2.d/S72inetsvc;**

Here we not only removed the old links but we added the new link to the modified inetsvc script. Don't forget to ensure the script is executable.

- Run **chown 744 /etc/init.d/inetsvc.modified**.

In the **/etc/init.d/syslogd** startup script, edit the command line options to the syslogd command. You should add a –t option. This option specifies that syslog will not listen to incoming messages from other servers.

Adding Network Configuration Parameters

We need to configure the network parameters as soon as possible after the network is brought up. Thus let's create a startup script call **/etc/init.d/nddconfig**. In it we will set all the relevant network parameters.

SYN flooding can tie up network resources by half-opening TCP sockets. Reduce problems associated with SYN floods with the following code. Tcp_conn_req_max_q0 is the number of half-open tcp connections allowed. Tcp_ip_abort_cinterval is the amount of time (measured in 1/1000 seconds) each connection is allowed to stay in a half-open state. Here we allow for 8192 half open connections in the queue. Each connection is allowed to remain half-open for 60 seconds.

- **/usr/sbin/ndd -set /dev/tcp tcp_conn_req_max_q0 8192**
- **/usr/sbin/ndd -set /dev/tcp tcp_ip_abort_cinterval 60000**

Get rid of smurfing and block mapping like this. Here we disable

ip_respond_to_timestamp and ip_respond_to_timestamp_broadcast that would normally allow the system to respond to ICMP timestamp requests. That might allow an attacker to learn the system's time. We also disable ip_respond_to_timestamp_broadcast that would normally allow an attacker to use an ICMP query to learn the netmask of the system. The ip_forward_directed_broadcasts option that would normally allow a system to route broadcast packets to a local network is also disabled.

- **/usr/sbin/ndd -set /dev/ip ip_respond_to_timestamp 0**
- **/usr/sbin/ndd -set /dev/ip ip_respond_to_timestamp_broadcast 0**
- **/usr/sbin/ndd -set /dev/ip ip_respond_to_address_mask_broadcast 0**
- **/usr/sbin/ndd -set /dev/ip ip_forward_directed_broadcasts 0**

Decreasing the amount of time arp information remains in the arp cache can help alleviate some spoofing attacks. Here is the method we are going to use to improve the arp timeouts on this server.

- **/usr/sbin/ndd -set /dev/arp arp_cleanup_interval 60000**
- **/usr/sbin/ndd -set /dev/ip ip_ire_arp_interval 60000**

We can disable ICMP redirects in order to help prevent denial of service attacks. Redirects are ICMP messages that a system can receive and change it's route table based upon those messages. Here we disable listening to those messages and sending them.

- **/usr/sbin/ndd -set /dev/ip ip_ignore_redirect 1**
- **/usr/sbin/ndd -set /dev/ip ip_send_redirects 0**

Source routing allows an attacker to implement their routing on your network. In this case the packet itself contains the route it should take. It is bad to let anyone do this. Turn off source routing like this.

- **/usr/sbin/ndd -set /dev/ip ip_forward_src_routed 0**

Ensure IP Forwarding is off. This way the system cannot be used as a router. This is turned off other places, but we will just make sure. Here we also turn on strict destination multi-homing which means if we had two network interfaces, we could not route to interface B through interface A and vice-versa.

- **/usr/sbin/ndd -set /dev/ip ip_forwarding 0**
- **/usr/sbin/ndd -set /dev/ip ip_strict_dst_multihoming 1**

Once this is all inside **/etc/init.d/nddconfig**, verify the file is executable. Create a symbolic link from /etc/rc2.d which will place to run just after the inet startup scripts.

- **chmod 744 /etc/init.d/nddconfig**.

- **cd /etc/rc2.d ; ln –s ../init.d/nddconfig ./S73nddconfig**

Additional Network Configuration

Create an **/etc/notrouter** file. This is more insurance that the system not to act as a router. Create **/etc/defaultrouter** and add the IP address of the router or firewall which will do all of this server's routing. The existence of **/etc/defaultrouter** file is also an indicator not to start route daemons.

Alternative to Running Sendmail as a Daemon

Also, we need to configure sendmail to run in cron every hour to get rid of mail that is queued up. Add this entry to root's crontab file. This is in lieu of having sendmail running as a service. Since we are not receiving mail, we do not need the service.

- **0 * * * * /usr/lib/sendmail –q**

Alternatively, we could run sendmail as a daemon that clears out the queue every hour and does not listen to port 25. Since we removed this script already, the option is presented as information only.

- Edit **/etc/init.d/sendmail** script to start sendmail using **sendmail –q1h**

Additional Logging with Syslogd

We need to configure some additional logging in the **/etc/syslog.conf** file. Create an entry for auth.info to go somewhere.

- **auth.info /var/log/authlog**

A tab must separate the entries. You must create the authlog file in order for this entry to take effect.

- **touch /var/log/authlog;**
- **chown root /var/log/authlog;**
- **chmod 400 /var/log/authlog;**

Prevent ISN Attacks

Setup the system to conform to RFC 1948 concerning initial sequence number generation. Edit **/etc/default/inetinit**, changing to **TCP_STRONG_ISS=2**. This will aid in preventing an intruder from guessing sequence numbers that might allow him or her to steal connections.

Unnecessary User Accounts

We can get rid of some of the default unnecessary user accounts. We definitely won't be using uucp. We do not need the listen user, nor do we need nobody4.

- **userdel uucp**
- **userdel nuucp**
- **userdel listen**

We can also be extra careful and disable logging in for the other default users.

- **passmgmt –m –s /dev/null daemon;**
- **passmgmt –m –s /dev/null bin;**
- **passmgmt –m –s /dev/null adm;**
- **passmgmt –m –s /dev/null lp;**
- **passmgmt –m –s /dev/null nobody;**
- **passmgmt –m –s /dev/null noaccess;**

Disabling Cron for All Users Except Root

We do not need any users other than root running cron jobs. Inside the /etc/cron.d directory, remove the cron.deny and at.deny files. Then we should add cron.allow and at.allow. Each allow file should have the single entry 'root'. Here is how we accomplish this.

- **cd /etc/cron.d;**
- **rm cron.deny at.deny;**
- **echo "root" > cron.allow;**
- **echo "root" > at.allow;**
- **chmod 400 cron.allow at.allow;**
- **chown root:root cron.allow at.allow;**

Prevent Buffer Overflow Attacks

Edit the **/etc/system** file. Add entries for **noexec_user_stack=1** and **noexec_user_log=1**. This will aid in preventing the famous, dreaded and nasty buffer overflow attacks where users send invalid input to programs in an effort to get the programs to execute malicious code. The appropriate lines would be…

```
set noexec_user_stack=1
set noexec_user_log=1
```

Configuration of the DNS Client

Now it's time to configure the DNS client. This is very simple. You need to know your domain name and the IP address of the DNS server. Once you have that do the following.

- Edit **/etc/resolv.conf** to include **nameserver x.x.x.x** for each nameserver you have. Your resolv.conf should look something like this.

```
nameserver 192.168.10.12
nameserver 192.168.20.12
nameserver 192.168.30.12
```

- Edit **/etc/nsswitch.conf**, ensure that the line containing the **hosts:** entry reads **hosts: files dns**. This tells the name services client to check the hosts file then DNS when looking for name resolution. It should now look something like this.

```
passwd:     files
group:      files
hosts:      files dns
networks:   files
protocols:  files
rpc:        files
ethers:     files
netmasks:   files
bootparams: files
publickey:  files
netgroup:   files
automount:  files
aliases:    files
services:   files
sendmailvars:   files
```

Note we install DNS for extra logging information. But, DNS does not necessarily provide a trustworthy entry in your log files. It often provides convenient and helpful additional information, in critical security situations you should also double check the IP addresses in log files.

Configuration of the NTP Client

We need to configure the NTP client. This is a simple but important task. It is important that system time is accurate for any auditing that may be necessary. Let's assume there are three intranet timeservers that we can get time from (192.168.10.99, 192.168.20.99 and 192.168.30.99). We want to edit the **/etc/inet/ntp.conf** file accordingly. It should look

something like this.

```
driftfile /etc/ntp.drift

server 192.168.10.99
server 192.168.20.99
server 192.168.30.99

restrict default ignore

restrict 192.168.10.99 nomodify
restrict 192.168.20.99 nomodify noquery
restrict 192.168.30.99 nomodify noquery
```

Here we have configured NTP not to listen to anyone but the three internal servers that provide time. For debugging purposes we allow querying from 192.168.10.99. This may be a bit overzealous given they are all internal servers, but we have erred on the side of caution in this case. Once you have finished ntp.conf setup you can start the xntp daemon using **/etc/init.d/xntpd start**. You might note that if your system time is drastically off it will be corrected at this time, as the start script runs **ntpdate** to set the clock to the timeservers' time.

Configuring Auditing and Accounting

The accounting packages have been installed already. We just need to uncomment a few lines.

- Edit **/etc/init.d/perf** and uncomment lines indicated in the script.
- Edit **/var/spool/cron/crontabs/sys** as follows. Then send a HUP signal to cron. Which you can do by finding the process id of cron with **ps –ef | grep cron | grep –v grep | awk '{print $2}'** then send it a signal by using **kill –HUP *cron's_pid***.

```
0,15,30,45 * * * * /usr/lib/sa/sa1
50 23 * * * /usr/lib/sa/sa2 -s 00:00 -e 23:59 -I 1200 -A
```

- We now need to modify **/usr/lib/sa/sa1** and **/usr/lib/sa/sa2** to include year and month in the output file names. To do this we change the **DATE=`/usr/bin/date +%d`** to **DATE=`/usr/bin/date +%Y.%m.%d`**. We need to make this change in both scripts.
- Also, in **/usr/lib/sa/sa2** we should comment out the **find** command at the end of the script that deletes the output files from accounting. Keep in mind you will need to watch file system space, eventually.

Consider using the features of process accounting included in the accounting packages. If

this is a highly used system it may be hindered by the overhead of writing to a file every time a process ends. On the other hand process accounting can provide very useful information, particularly if you are unsure what is happening on the system.

- Enable process accounting with **/usr/lib/acct/accton /var/adm/pact**. Here **pacct** is the output file containing the process accounting information.

Go Away Messages and Other Miscellaneous Configurations

Create **/etc/issue** and **/etc/motd** files. The issue file is for users before they login. The motd files is for users directly after they login. In them put some sort of message telling users that their accesses will be appropriately logged. Also, if they are not supposed to be users on that system then they are not welcome to login. See your company's lawyer for the best wording for these messages. Note there is no telnet login so this will really only be printed on the console.

Installation and Setup of SSH

Create a CD-ROM or tape containing the latest version of SSH. At the time of this document I am using version 3.0.1. Do not use 3.0.0 if you expect to use password authentication at all. There is a serious but in 3.0.0 regarding password authentication. If you don't want to compile them yourself you can get them from http://www.ssh.com/. In this situation I recommend compiling the SSH binaries yourself because you can configure the included options to sshd. Since we are looking specific behavior here is the method we will use to compile SSH.

- **gunzip –c ssh-3.0.1.tar.gz | tar xvf –**
- **cd ssh-3.0.1**
- **./configure –enable-debug –disable-X11-forwarding –disable-tcp-port-forwarding**
- **make**
- **make install**

Here we have chosen to enable debugging, disable X11 forwarding and disable tcp-port forwarding. Debugging can be helpful in the chroot environment setup and if there are any problems later. The X11 forwarding option aids in tunneling X window traffic. Our users have no use for X window traffic to or from this server. Tcp port forwarding does just what it sounds like. With tcp port forwarding, a user could listen on a local port and push that traffic back to their client. Likewise, a user could push traffic from their client and make it look like it was originating from our server.

Create the host keys. This can be done by using **ssh-keygen –P /etc/ssh2/hostkey**

Edit the **/etc/ssh2/sshd2_config** file. Ensure **VerboseMode** is set to **no**. **CheckMail**

should be set to **no**. You do not want SSH doing anything additional for the user. **UserConfigDirectory** can be reset to indicate a directory not inside the user's home directory (**"/etc/ssh2/user_auth/%U"** for example). You should also set PermitRootLogin, AllowX11Forwarding and **AllowTcpForwarding** to **no**. We already addressed tcp and X forwarding in the compilation stage but we'll be extra cautious. This is also here in case you were lazy or didn't have a compiler (please don't be lazy).

Create an SSH startup script in /etc/init.d/sshd2. Here is one with no bells or whistles.

```
#!/bin/sh

umask 077
case "$1" in
  'start')
      /usr/local/sbin/sshd2
      ;;
  'stop')
      ps -ef | grep "/usr/local/bin/sshd2" | grep -v grep |
awk '{print $2}' | xargs kill
      ;;
esac
```

Now just make links in the rc2.d and rc3.d directories. You can do this with the following code.

- **ln –s /etc/init.d/sshd2 /etc/rc2.d/K75sshd;**
- **ln –s /etc/init.d/sshd2 /etc/rc3.d/S75sshd**.

Now the SSH daemon is setup for our environment. Below we will be configuring the users.

Setup User Environment Explanation

We are going to create a restricted shell chrooted user environment. We only want to give the users access to commands necessary to remotely run sftp and scp. Users will not have access to change their own home directories nor will they be allowed to administrate their own public key. A public key will need to be submitted to the system administrator in order for the user to gain access. The user will have write access to a subdirectory of their home directory to place their files in. The purpose of this is for the user to be able to securely drop off files or pick up files. These files either need to be transferred to or from someone internal to the company.

Setting Up the User's Home Directory

We must create all files necessary for the user inside their home directory, including devices, library files and executables. They will not have access to any other files on the file system.

First, we will create the user. Let's call the user **user**. Here is how, **useradd –d /home/user user**. Give the user a password with **passwd user**. This password will never be used, as we have not setup any services that allow password authentication. Now that the user exists let's setup a home directory.

mkdir –m 755 /home/user
mkdir –m 755 /home/user/usr
mkdir –m 755 /home/user/dev
mkdir –m 755 /home/user/usr/bin
mkdir –m 755 /home/user/usr/lib
mkdir –m 755 /home/user/usr/share
mkdir –m 755 /home/user/usr/share/lib

Now let's setup the files inside our directory structure. We'll start with the timezone information.

cd /usr/share/lib;
tar cf - zoneinfo | (cd /home/user/usr/share/lib; tar xfp -);

Setting up device files is very important. You will need these files in order for networking to work.

cd /home/user/dev;
ls –lL /dev/tcp /dev/ticotsord /dev/udp /dev/zero

Record the information from this command about major/minor numbers and create the device files using the mknod command. For example:

crw-rw-rw- 1 root sys 42, 0 Aug 13 18:45 /dev/tcp
mknod c 42 0 tcp

Copy binaries for ls, mv, rm, and sftp-server2 to the /home/user/bin directory.

cp /usr/bin/ls /home/user/usr/bin/
cp /usr/bin/mv /home/user/usr/bin/
cp /usr/bin/rm /home/user/usr/bin/
cp /usr/local/bin/sftp-server2 /home/user/usr/bin/
cd /home/user/usr/bin ; ln –s sftp-server2 sftp-server
cd /home/user ; ln –s usr/bin bin

In the same manner, copy the libraries needed for these binaries.

cp /usr/lib/libc.so.1 /home/user/usr/lib/
cp /usr/lib/libcurses.so.1 /home/user/usr/lib/
cp /usr/lib/libdl.so.1 /home/user/usr/lib/
cp /usr/lib/libgen.so.1 /home/user/usr/lib/
cp /usr/lib/libm.so.1 /home/user/usr/lib/
cp /usr/lib/libmp.so.1 /home/user/usr/lib/
cp /usr/lib/libmp.so.2 /home/user/usr/lib/
cp /usr/lib/libnsl.so.1 /home/user/usr/lib/
cp /usr/lib/libsec.so.1 /home/user/usr/lib/
cp /usr/lib/libsocket.so.1 /home/user/usr/lib/
cp /usr/lib/nss_compat.so.1 /home/user/usr/lib/
cp /usr/lib/nss_dns.so.1 /home/user/usr/lib/
cp /usr/lib/nss_files.so.1 /home/user/usr/lib/
cp /usr/lib/nss_nis.so.1 /home/user/usr/lib/
cp /usr/lib/nss_nisplus.so.1 /home/user/usr/lib/
cp /usr/lib/rsh /home/user/usr/lib/
cp /usr/lib/straddr.so /home/user/usr/lib/
cp /usr/lib/straddr.so.2 /home/user/usr/lib/

Let's create a .hushlogin file in the user's home directory to prevent /etc/profile from outputting unnecessary verbiage or executing unnecessary programs.

touch /home/user/.hushlogin

Now we need to make sure the user can't write to any of these files except the device files.

cd /home/user;
chown –R root:root .
chmod –r a-w .
chmod 666 dev/*

Now we just need a single write enabled directory to allow the user to put their files in.

mkdir depot
chown user depot
chmod 755 depot

That does it for the home directory. As a note, it can be time consuming to track down all the necessary files for a chroot solution. It is important to use only the files that are essential inside the chrooted environment. Here is what I did to get this chrooted environment working.

- Try to find out what shared object libraries sshd uses. Start with **ldd /usr/local/sbin/sshd** to list all the shared object libraries for the sshd executable. Copy those from the system library directory (/usr/lib) to **/home/user/usr/lib/**.
- Run sshd in debug mode (use **sshd –D 2**).
- Try to connect as the user you have setup.
- Check for output messages. Possibilities are lib_ld.so.1 messages complaining about not being able to open a system library. In that case copy the named library to the chrooted environment. You might run into a problem where other types of files do not exist. Find them and figure out if they are really needed. After each fix, try again with the debug option. This should take care of most if not all of the problems your run into.
- If you get through all of the debug messages and still can't figure out what is breaking you could you the truss command. For example, **truss –p {sshd's_pid} –rall –wall –f –o /tmp/outputfile**. This generates a ton of output and you will probably have to spend some time looking at it to find problems, but it is very effective.

A Simple Chroot Solution

Here I will provide a simple example of how to chroot an environment. The example includes some bad things like suid-root shell scripts and not changing the setuid after the chroot call. I would encourage anyone attempting this for real to write his or her own shell in a compiled programming language like C. With that said, the following example gives you the basic steps to get it done. Create an executable file in say **/usr/local/bin/chroot_rsh**. Make the mode 555. For an example or more information on using chroot the proper way, please see Aaron Gifford's web page listed in this paper's bibliography.

```
#!/bin/ksh

# Clean up the environment.
unset IFS
export PATH=/bin:/usr/bin
export LD_LIBRARY_PATH=/usr/lib

if [[ "$1" == "" ]]
then
  # Don't allow ssh without a command attached.
  exit 1
else
  cd $HOME
  /usr/sbin/chroot $HOME /usr/lib/rsh $@
fi
```

Change the user's shell to **/usr/local/bin/chroot_rsh**. You can use **usermod –s /usr/local/bin/chroot_rsh user**. Or just edit the password file.

User Setup - Adding Public Keys and User Environment

Users must create their own public private key pair and send you the public key. They should be strongly encouraged to set a good pass phrase for their private key. Mention that they don't have to be restricted to eight characters. Once you have received the public key put it in the directory you setup for keys. I hate to have to put this in here, but if they send you the private key or the public-private key pair, get rid of them and explain that they need to generate a new pair and just send the public key. It could happen. Earlier we setup **/etc/ssh2/user_auth/%U** as for our keys. So now we want to create the **/etc/ssh2/user_auth/user** directory. Inside that directory we need an **authorization** file, which will contain the name of the public key file. We also need the public key.

- **echo "Key public_key_name.pub" > /etc/ssh2/user_auth/user/authorization;**
- **chmod 644 /etc/ssh2/user_auth/user/authorization;**
- **mv public_key_file.pub /etc/ssh2/user_auth/user/public_key_name.pub;**
- **chmod 444 /etc/ssh2/user_auth/user/public_key_name.pub;**

Also, you don't want users messing with their PATH variable. You need to set this in a file called environment inside the user configuration directory.

- **echo "PATH=/bin" > /etc/ssh2/user_auth/user/environment;**
- **chmod 444 /etc/ssh2/user_auth/user/environment;**

That's all there is to it. It is important to keep a close eye on which users are using what keys. Expiration is a very good idea. You never know who is going to leave a company and accidentally take a few keys with them.

Good Now Do It Again! (A.K.A. Backups)

Once you have everything setup in way you know to be sane and secure, back it up. Back it up at least two times, maybe three when tapes are cheap. Send one tape offsite. Here is a procedure for doing the backup.

- Bring up the system in single user mode. Run **init 0** then when at the ok prompt **boot –s**.
- Mount all the file systems using the **mountall** command.
- Insert tape.
- Rewind the tape with **mt /dev/rmt/0 rewind**
- Check the tape/drive status with **mt /dev/rmt/0 status**.
- Backup each file system using **ufsdump**.

    - **ufsdump /dev/rmt/0n /**
    - **ufsdump /dev/rmt/0n /var**
    - **ufsdump /dev/rmt/0n /opt**
    - **ufsdump /dev/rmt/0n /usr**
    - **ufsdump /dev/rmt/0n /home**
- Eject tape (**mt –f /dev/rmt/0 rewoffl**) and repeat the procedure with another tape.

These tapes now contain valuable system configuration information. Make sure they are stored securely and in different locations in case of a disaster. They can also come in handy when an executive inevitably wants another system to do just what this one does somewhere else on the network. Be a hero. Make backups. Ok, enough of that, you get the point.

Go Live! (Almost)

Plug in the network cables. Verify the simple services like DNS, NTP, default routing, and outbound sendmail work. Use **/usr/sbin/ping *hostname*** to verify DNS and routing work. You can check to see that NTP is working with the **ntpq –p** command. You might see output that looks like this…

```
     remote           refid      st t when poll reach   delay   offset    disp
==============================================================================
*timeserver1     128.252.19.1     2 u   48  512  377    15.46   -0.325    1.69
*timeserver2     128.252.19.1     2 u   79  256  336     3.59   -0.114    3.40
*timeserver3     128.252.19.1     2 u   44  256  336     3.99   -0.104    3.10
```

On the ball NTP administrators will quickly realize I changed a lot to protect the innocent here, but that's basically the format.

Also, spend some time trying to ssh in and break your security the way a user or an attacker might attempt. Try copying executables to the depot directory and executing them. If that works, then don't deploy the server.

Go Live! (Really)

Now you are ready. But that doesn't mean you can open the firewall add 3000 users and forget about the server (or worse never login again). You need to stay in touch with the system and what is normal activity on it. Check log files. Watch for capacity problems in disks and performance. Educate users as you add them. Create some documentation for users explaining how the system should work from their point of view and how to best help keep their files secure. Keep up with the latest patches for all the system software and operating system.

References

Steve Acheson. "The Secure Shell Frequently Asked Questions." URL: http://www.employees.org/~satch/ssh/faq/ssh-faq.html (September 16, 2001)

Lee Brotzman and Hal Pomeranz. "Track 6—LevelTwo Securing UNIX 6.4 Running UNIX Applications Securely." Baltimore: SANS 2001, May 2001.

Lee Brotzman and Hal Pomeranz. "Track 6—LevelTwo Securing UNIX 6.5 Linux/Solaris Practicum." Baltimore: SANS 2001, May 2001.

Jeff Campione. "Solaris 8 Installation Checklist." URL: http://www.sans.org/y2k/practical/Jeff_Campione_GCUX.htm (September 16, 2001)

Simson Garfinkel and Gene Spafford. Practical UNIX & Internet Security. Sebastopol: O'Reilly, 1996.

Aaron Gifford. "CHRSH: A chroot jail wrapper for ordinary Unix shells." URL: http://www.aarongifford.com/computers/chrsh.html (September 16, 2001)

Stuart McClure, Joel Scambray, and George Kurtz. Hacking Exposed. Berkeley: Osborne/McGraw-Hill, 1999.

David Mills. "Access Control Options." URL: http://www.eecis.udel.edu/~ntp/ntp_spool/html/accopt.htm (September 16, 2001)

Hal Pomeranz. "Solaris Security Step by Step. Version 2.0." SANS Institute. 2001.

SSH Communications Security. "SSH Secure Shell for UNIX Servers Quick Start Guide." URL: http://www.ssh.com/support/ssh/ssh2-quickstart.pdf (September 16, 2001)

Keith Watson. Sun Microsystems. "nddconfig." September 29,1999. URL: http://www.fish.com/titan/arch/sol2sun4/lib/nddconfig (September 16, 2001)