SANS
Global Information Assurance Certification (GIAC)
Program

GCUX
Certified Unix Security Administrator
Version 1.8

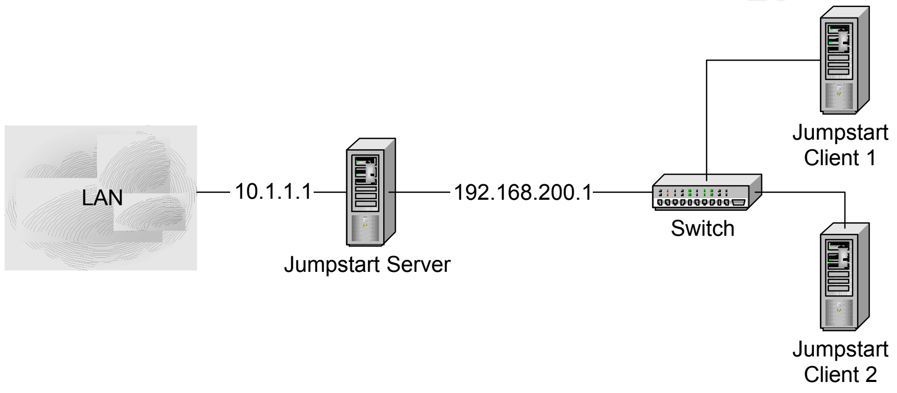
# Building and Securing a Solaris 8 Jumpstart Server

By
Michael J. Huffner
December 2001

## Table of Contents

## System Description

When building and securing systems manually, how often are steps missed? By using Jumpstart, these mistakes can be prevented. Jumpstart is a Solaris utility that automates the installation of the operating system and any additional software. Using Jumpstart allows all systems to be built exactly the same and the opportunity for human error to be greatly decreased. The purpose of this Jumpstart server is to install the operating system on a new system, secure the system, and install additional packages that allow secure access and monitoring of the new system. Here is a diagram of the proposed configuration:



The LAN IP address of the Jumpstart server is 10.1.1.1 and the private interface has an IP address of 192.168.200.1/24. Ideally the Jumpstart server will be placed on the test/development network that does not have Internet access and does not have access to the WAN. While this does not protect the system from attacks, it does limit the number of places the attacks can originate.

### Hardware

The system used for these setup procedures is a Sun Enterprise 220R with dual UltraSPARC-II 450MHz processors, 1GB RAM, 2 x 18 GB hard disks, and one additional 10/100 Ethernet adapter (hme). The additional network card is for jumpstart use only. That interface will either be used with a crossover cable directly connected to the client system or plugged into a switch that has only systems that require installation. This hardware is extreme overkill for a system that acts solely as a jumpstart server. The recommended minimum to install this system is an Ultra Enterprise 1 with an UltraSPARC processor, 256MB RAM, and two 4GB disk drives, one if disk mirroring is not going to be used. Use of a tape drive is also highly recommended. The hostname of the system in this example is loki. The server has a “public” IP Address on hme0 of 10.1.1.1. The second interface (hme1) is for jumpstarting new systems only and has an IP Address of 192.168.200.1. The entry in the hosts file for that address is loki-js.

**Software**

In addition to Solaris 8, the following software is being used:

| Software | Version | Homepage |
|---|---|---|
| md5 | 6142000 | http://www.sunfreeware.com/programlistsparc8#md5 |
| Binutils | 2.11.2 | http://www.gnu.org/order/ftp.html |
| Lsof | 4.60 | ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/ |
| Sudo | 1.6.3p7 | http://www.courtesan.com/sudo/ftp.html |
| GNUPG | 1.0.6 | http://www.gnupg.org/download.html |
| IP Filter | 3.4.20 | http://coomds.anu.edu.au/~avalon |
| Logcheck | 1.1.1 | http://www.psionic.com/abacus/logcheck |
| Tripwire | 1.3.1-1 | http://www.tripwire.com/downloads/tripwire_asr/ |
| Snort | 1.8.3 | http://www.snort.org/downloads.html |
| snort rules | Current | http://www.snort.org/downloads.html |
| RPCBind | 2.1 | ftp://ftp.porcupine.org/pub/security/index.html#security |
| Tara | 2.0.9 | http://www-arc.com/tara/index.shtml |
| Gcc | 3.0.2 | http://www.gnu.org |
| Gcc | 2.95.3 | http://www.gnu.org |
| Nmap | 2.54 Beta 22 | http://www.nmap.org |

NOTE: IP Filter 3.4.22 is the current version at the time of this document. However, there is an issue with the software causing the operating system to freeze without reporting any error messages. According to the IP Filter mailing list, (http://false.net/ipfilter/2001_12/), there can be some issues with Solaris 8 and the current version. Reverting back to version 3.4.20 seems to solve these problems.

## Risk Analysis

Even though this system has no "sensitive" data, it needs to be very secure. If an attacker gains root access to this system, they have gained quite a bit of information about all the systems in the environment. The attacker could use this information to determine which exploits to use to break into other systems. Another problem is that the attacker could modify the jumpstart configuration and have a Trojan horse installed on every system that has been built from that point on.

### Physical Access

As stated in the SANS Solaris Practicum, physical access equals root.[1] If an attacker gains access to this system, then the attacker will know the basic configuration of all servers built with this server. Access can be gained by powering off the system a number of times until the root file system becomes corrupt. Once this happens the system will boot into single user mode and not prompt for a password. Another way for an attacker to gain access would be to power off the system, boot the system with just the Solaris Operating System install, mount any of the partitions, and from there any number of things can be done. (Trojan horse, change root password, etc.)

### TFTP

Since the new server doesn't have an operating system, one will have to be provided via tftpboot. TFTP is notoriously buggy and usually the /tftpboot directory contains interesting files. In this case it only contains a boot file for Solaris, however the port is listening for connections and the owner of the service is root. If vulnerability is exploited, an attacker could possibly obtain root access.

### Remote Procedure Calls (RPC)

NFS is required for jumpstart to work. NFS requires RPC, which also has a history of buggy implementations. The way RPC works also makes it difficult to firewall. At boot, the RPC service binds to a random port and then registers that information with rpcbind, which listens on port 111. A client makes a request to rpcbind to find out which port the desired service is listening on. The client then starts communicating with the desired service on that port. Since the RPC bind uses a random port determined at boot time, there is no way to predict what port needs to be blocked.

### NFS

The software via jumpstart cannot be installed if it cannot be mounted. So NFS is required. NFS has the same history of security problems as RPC and TFTP. Also, NFS can also be configured and administered poorly. There are a number of exploits against NFS: file handle, user's exported home directories, creating device files via NFS, etc.

Implementing a host-based firewall can solve the majority of these issues. This prevents undesired connections while allowing the appropriate traffic to connect. On the LAN interface, only ssh traffic is permitted. While on the private network, clients are allowed to communicate with the jumpstart server with the necessary protocols. To help protect against TFTP vulnerabilities, the /tftpboot slice is mounted read-only and no set-uid. This prevents attackers from uploading executables via TFTP that are set-uid root.

## Jumpstart Server Installation

### Physical Preparation

The first step to build this server is to physically secure the system by placing it in a secure area. If the system is in a computer room where a number of people have access, place the system in a locked cabinet - preferably one with metal doors. Attach any peripherals and install any additional parts, such as RAM, etc. Next, connect the system to a console server or any other device that will give you terminal access to the system. Finally make sure the system is not connected to the network.

### Operating System Install

Before starting the installation process, please be sure to read any and all documentation that came with the media. Insert the proper CD-ROM and begin the installation. Follow the installation and answer the questions accordingly.

| Prompt | Answer |
|---|---|
| Is the machine networked? | Yes, continue and enter IP Address, netmask, default route, and no to name service. |
| Time Zone: | If systems are spread out over multiple time zones, consider using GMT. This can make it easier to track events when comparing logs. |
| Enter root password: | Enter a password. Please be sure not to use any dictionary words. |
| Install Type | Custom, only the Core System Support will be installed. |
| Additional Products | Deselect all. These language packages are not needed. |
| Software Group | Core System Support Only. |

Next, manually layout the disk configuration. Note that Solaris 8 does not require that swap space be at least equal to the amount of RAM. However, if disk space permits, it cannot hurt for the two to be equal.

| Slice | File system | Size |
|---|---|---|
| s0 | / | 300 MB |
| s1 | Swap | 1-2x Physical Memory |
| s2 | Backup | - |
| s3 | /usr | 1024 MB |
| s4 | /var | 2048 MB |
| s5 | /opt | Rest |
| s6 | /tftpboot | 10 MB |
| s7 | /spare | 10 MB |

The file system /spare can be omitted if disk mirroring via DiskSuite is not going to be used. If possible, consider making /var even larger, that way even more logging information can be retained. /opt is to be used for any third party applications. The /tftpboot partition is used for the jumpstart application and eventually that slice will be mounted read only.[2] This is because tftp is notorious for being "buggy" and having numerous security holes. At this time you can configure any other disks if you wish. If not, confirm and continue. The system will start the install and will reboot afterwards.

Not everything needed is installed with the Core Group, so these additional packages will have to be installed manually. These packages can be found on the installation media. Again, if disk mirroring is not being used, skip the installation of the Solstice Disksuite packages.

| Software | Packages Required |
|---|---|
| Utility: showrev | SUNWadmc SUNWadmfw |
| NTP | SUNWntpr SUNWntpu |
| Perl | SUNWlibms SUNWpl5p |
| System accounting | SUNWaccr SUNWaccu |
| Solstice Disksuite | SUNWmdr SUNWmdu SUNWmdx |
| Gzip | SUNWgzip |
| Source Compiling | SUNWhea SUNWarc SUNWarcx SUNWsprot SUNWsprox SUNWbtool SUNWbtoox SUNWtoo SUNWtoox SUNWhmdu SUNWtnfc SUNWtnfcx SUNWtnfd SUNWlibm SUNWlibC SUNWlibCx SUNWlilbCf SUNWcstl SUNWcstlx SUNWscpu SUNWscpux |
| Jumpstart | SUNWinst<br>Note: pkgadd will state that SUNWoldte and SUNWolrte are prerequisites. That message can be ignored. |
| Custom Jumpstart scripts | SUNWxcu4 SUNWxcu4x |
| Terminal Support | SUNWter |

After all Solaris packages have been installed, it is now time to install the latest patches. The latest patches and md5 checksum can be retrieved from sunsove.sun.com via ftp. The files are named 8_Recommended.zip and CHECKSUMS and can be found in /pub/patches. Prior to installing the patches, compare the md5 checksum against the entry in the CHECKSUMS file. Use something other than the network to get the patches over to the current system (Tape, CD-ROM.) Create the /opt/jumpstart/patches directory and copy the patch cluster there. These patches will also be used when jumpstarting a client. Unzip the patches and make sure to read any text documents that came with the patches. Install the patches and note that return code two means the patch is already installed and eight means that the package needing patched is not installed on the system. After the patches have been installed, check the log file of the installation for any other errors. The name and location of the log file are displayed in a message upon completion of the patch installation. Once the patches are installed, a reboot is required. After the system has rebooted, configure DiskSuite to mirror the Operating System disks if desired. For directions on how to perform this task, please see:

http://docs.sun.com/ab2/coll.260.2/DISKSUITEREF/@Ab2TocView?Ab2Lang=C&Ab2Enc=iso-8859-1
Be sure to remove /spare from /etc/vfstab and use that for DiskSuite.

## Hardening the Operating System

In many vendors' infinite wisdom, the operating system by default is wide open. Sun is no exception to that rule. Now it is time to close these many holes. First disable all unnecessary services. In /etc/rc2.d run:

```
# for file in S30sysid.net S71ldap.client S71sysid.sys S72autoinstall \
>         S73nfs.client S74autofs S76nscd S80PRESERVE
> do
> ./$file stop
> mv $file .off.$file
> done
```

This turns off all of the above services and prevents them from starting up when the system boots up. Next set the umask for system daemons at startup so that log files created by these daemons are created with the desired permissions. The setting is 022 by default. This gives everyone the ability to read the files. If paranoid, set the values of CMASK in /etc/default/init to 077 and only root can access the files. Also, for something in the middle-the-road paranoia, the setting 027 gives the group permissions of read while preventing everyone else from accessing the files.[3]

The initial network configuration upon boot is started by /etc/rc2.d/S69inet. There are a number of security related configuration changes we need to make as soon as possible after the initial network configuration has finished. Create a script in /etc/init.d called netconfig and it should look like:

```
#!/bin/sh
# syn floods
/usr/sbin/ndd -set /dev/tcp tcp_conn_req_max_q0 8192
/usr/sbin/ndd -set /dev/tcp tcp_ip_abort_cinterval 60000
# mapping and smurfing
/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
/usr/sbin/ndd –set /dev/ip ip6_respond_to_echo_multicast 0
#  arp spoofing
/usr/sbin/ndd -set /dev/arp arp_cleanup_interval 60000
# ignore redirects
/usr/sbin/ndd -set /dev/ip ip_ignore_redirect 1
/usr/sbin/ndd -set /dev/ip ip6_ignore_redirect 1
/usr/sbin/ndd -set /dev/ip ip_send_redirects 0
/usr/sbin/ndd -set /dev/ip ip6_send_redirects 0
# disable source routing
/usr/sbin/ndd -set /dev/ip ip_forward_src_routed 0
/usr/sbin/ndd -set /dev/ip ip6_forward_src_routed 0
# disable IP forwarding
/usr/sbin/ndd -set /dev/ip ip_forwarding 0
/usr/sbin/ndd -set /dev/ip ip6_forwarding 0
/usr/sbin/ndd -set /dev/ip ip_strict_dst_multihoming 1
/usr/sbin/ndd -set /dev/ip ip6_strict_dst_multihoming 1
```

The first group of rules sets the maximum number of connection requests to 8192 and sets the timeout sixty seconds. By increasing the queue size and decreasing the timeout, the system is better prepared against SYN Floods. The second set of rules helps to protect against mapping and Smurf attacks. The first three entries of the group will cause the system to not respond to particular ICMP messages, which are not normally used. The final entry of the group prevents multi-homed systems from passing the broadcast traffic to other interfaces. The next entry helps to lessen the effects of an arp DOS attack by setting the timeout to sixty seconds. The next set of entries is used to ignore ICMP redirect packets. This prevents the system from being pointed to another gateway. If this is not in place, an attacker could use this as a DOS attack or an attempt to intercept the traffic. The next entry is used to disable source routing. Source routing may be used to bypass firewalls, access control lists, and network intrusion detection systems and route the traffic elsewhere. The final group is for multi-homed systems. The first entry prevents the system from acting as a router and sending packets between the two networks to which it is attached. The final entry catches any systems that may ignore the disabled ip_forwarding entry.[4] IPv6 entries were put in place even though it is not used in this scenario. This is just in case IPv6 is enabled on the network, this system will already be secured against any IPv6 exploits.[5] Also create /etc/notrouter, this prevents routing daemons from starting up and causes the boot process to disable IP forwarding. This is also a good time to put the default route in /etc/defaultrouter (If this wasn't already entered during the installation process.) Now link /etc/init.d/netconfig to /etc/rc2.d/S69netconfig and these options will be set up the next boot. Set the owner and group of this script to both root, and the permissions should be set to 744. Test the new script by running /etc/rc2.d/S69netconfig just to make sure there are no errors.

Next create /etc/init.d/newinetsvc. This script disables DHCP support and multicast routing. This script was adapted from the SANS' Linux/Solaris Practicum and from the original script.[6]

```
#!/sbin/sh
#
case "$1" in
'start')
        ;; # Fall through -- rest of script is the initialization code
'stop')
        /usr/bin/pkill -x -u 0 'in.named|inetd'
        exit 0
        ;;
*)
        echo "Usage: $0 { start | stop }"
        exit 1
        ;;
esac

/usr/sbin/ifconfig -auD4 netmask + broadcast +

if [ -f /usr/sbin/in.named -a -f /etc/named.conf ]; then
        echo 'starting internet domain name server.'
        /usr/sbin/in.named &
fi

#mcastif=`uname -n`
#echo "Setting default interface for multicast: \c"
#/usr/sbin/route add -interface \
#       -n netmask "240.0.0.0" "224.0.0.0" "$mcastif"

# Run inetd in standalone mode (-s) and additional logging (-t)
/usr/sbin/inetd -s -t &
```

This script should be owned by root and have group also set to root. The permissions should again be 744. Next remove the original link /etc/rc2.d/S72inetsvc and create a new link for /etc/rc2.d/S72inetsvc pointing to /etc/init.d/newinetsvc.

The next step is to take syslog out of listen mode. This is achieved by adding the "-

t" switch when starting syslogd. Create a new syslog startup script by copying the original /etc/init.d/syslog to /etc/init.d/newsyslog and adding the "-t" to the line(s) where syslogd is started. Again, remove the link /etc/rc2.d/S74syslog and create the new link /etc/rc2.d/S74syslog pointing to /etc/init.d/newsyslog. Make sure root:root is the owner:group and the permissions are 744 on the new script.[7]

If there are no serial devices other than a terminal attached to the system, go ahead and remove the following line from /etc/inittab:

```
sc:234:respawn:/usr/lib/saf/sac –t 300
```

This disables the listener on any serial ports without disabling the login prompt on the console.[8]

Since no other services except for tftpd are being used, remove all other entries from the inetd.conf. This ensures that no other unused services are started and are listening for connections. Other files that can be removed since the services won't be used are **/etc/auto_***. Those files are used by the automounter and NFS. NFS is still usable by jumpstart, but the automounter has been disabled. Also remove useless cron jobs.[9]

```
# cd /var/spool/cron/crontabs
# rm adm lp
```

By default, sendmail is running in daemon mode listening on port 25. Disable this by performing the same type of tasks as above. Copy the original sendmail file in /etc/init.d to newsendmail. Change the line MODE="-bd" to MODE="". Remove the link /etc/rc2.d/S88sendmail and recreate the link pointing to /etc/init.d/newsendmail. Check that ownership and group is root and permissions are 744. This way mail will still be sent, but the server will no longer be listening whenever the system boots. Also it is a good idea to add "0 * * * * /usr/lib/sendmail –q" to cron.[10] This flushes the mail queue of any messages that were unable to be immediately sent.

**File System Security**

There are a few configuration changes that can be made to make the file systems more secure and faster system boots after crashes. The first step is to turn on logging for all file systems. (Except /tftpboot, it is too small) This is done by adding logging to the options portion of /etc/vfstab. This turns ufs into a simple journaling file system. This helps to prevent inconsistency after a crash and the system reboots faster after crashes because there is no need to run a file system check. A downside to enabling logging is if a system is doing a lot of disk writes, then there can be a performance hit.

Since /usr contains the majority of the binaries used by the Operating System, the file system should be mounted as ready only.[11] The reasoning being this is that hackers will store Trojan horses in /usr/bin and other places within /usr. Also, if a hacker does gain access to your system , they won't be able to modify any setuid files in /usr. A hacker would however be able to modify any setuid files in the file systems that are read/write.

By setting the remaining file systems to not allow setuid executables, this prevents hackers from running scripts that are setuid or hacking applications that have setuid executables.  Below is a copy of the /etc/vfstab used by this system. The columns are: device, raw device, mount point, file system type, fsck pass, mount at boot, and options.

```
fd                  -                   /dev/fd fd      -       no      -
/proc               -                   /proc   proc    -       no      -
/dev/md/dsk/d1      -                   -       swap    -       no      -
/dev/md/dsk/d9      /dev/md/rdsk/d9     /       ufs     1       no      logging
/dev/md/dsk/d3      /dev/md/rdsk/d3     /usr    ufs     1       no      logging,ro
/dev/md/dsk/d4      /dev/md/rdsk/d4     /var    ufs     1       no      logging,nosuid
/dev/md/dsk/d5      /dev/md/rdsk/d5     /opt    ufs     2       yes     logging,nosuid
/dev/md/dsk/d6      /dev/md/rdsk/d6     /tftpboot       ufs     2       yes   nosuid,ro
swap                -                   /tmp            -       yes     -
```

This protects the binaries in /usr while still allowing applications and other software to be installed onto /opt.

**Network Access**

To administer the system properly, network access is required. This can be accomplished by using Secure Shell (SSH). There are many flavors of SSH. This paper will focus on OpenSSH. OpenSSH allows the admin to securely perform administrative tasks. SSH encrypts the connection preventing the network traffic from snooping. Encryption also helps prevent session hijacking by making it more difficult. SSH also provides additional functionality by allowing the tunneling of other protocols such as X Windows and rsync. To compile OpenSSH, the software OpenSSL and zlib are required. Note that OpenSSL also requires Perl. OpenSSH can also be built to include support for TCP Wrappers, which allows for limiting what network or systems users can connect from. See the Additional Software section for how to install and configure TCP Wrapper, OpenSSL, and OpenSSH.

**User Access**

The more user accounts on a system, the more security holes there are in the system. Only users that need access to a system should have access to that system. For this jumpstart server, only system administrators that are tasked with installing new systems should have access.  Solaris 8 by default installs a number of users that are not needed.[12] To remove these users:

```
# for user in uucp nuucp smtp listen nobody4
> do
> /usr/sbin/passmgmt –d $user
> done
```

If process accounting is not going to be used, add the user **sys** to that list. Next set the default shell for other system users that are still needed.

```
# for user in daemon bin admin lp nobody noaccess
> do
> /usr/sbin/passmgmt –m –s /dev/null $user
> done
```

Even though the system is not running ftpd, it is a good idea to verify /etc/ftpusers exists and that system accounts are denied the ability use ftp. Solaris patches have been known in the past to turn on services that have been set to not run. The users root, daemon, bin, sys, adm, lp, uucp, nuucp, listen, nobody, noaccess, and nobody4 should be in the file.[13] This file actually prevents the specified users from logging in via ftp.

Next edit /etc/pam.conf to prevent login via rhosts. Remove the following lines:

```
rlogin   auth sufficient /usr/lib/security/$ISA/pam_rhosts_auth.so.1
rlogin   auth required /usr/lib/security/$ISA/pam_rhosts_auth.so.1
```

This prevents any users/hackers attempting to login via rhosts files.[14] Although ssh ignores this file, this prevents the files being used if for some reason another services is turned on. So now create empty root files for rhosts and other similar files and set the permission so that no one can access them.[15]

```
# for file in /.rhosts /.shosts /.netrc /etc/hosts.equiv
> do
> cp /dev/null $file
> chown root:root $file
> chmod 000 $file
> done
```

Finally for users we restrict cron usage to those who need it. In **/etc/cron.d/cron.allow** and **/etc/cron.d/at.allow** put an entry in for root only. The files should be read only for root and both owned by root. Also, if /etc/cron.deny exists, remove it. These entries only aloow root to modify cron jobs. Other users can still run cron jobs, they will just need a system administrator to modify or add jobs.[16]

## Logging and Accounting

### System Logs

Add at least one line to syslog.conf. Note that the white space must be tabs.[17]

```
auth.info	/var/log/authlog			# To log file
auth.info	@somehost				# To syslog server
auth.info	/dev/term/a				# To printer
```

This will log all messages except for debug to the desired destination. Also, LOG_AUTH must be specified in the syslog.conf:

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

Create the authlog file, make sure the owner is root, and set the permissions to 600. Also it is a good idea to create loginlog file in /var/log. This log file contains information from **login** such as failed attempts, etc.[18]

```
# touch /var/log/authlog
# touch /var/log/loginlog
# chown root /var/log/authlog /var/log/loginlog
# chmod 600 /var/adm/loginlog
```

One thing is certain about log files, they will grow. To prevent them from getting out of hand in size, the files should be rotated on a regular basis. The rotation period for this server will be one week. Depending on the amount of logging that is done, this can be increased or decreased. The more logging, the more often the files should be rotated. Below is the script being used to rotate logs on the jumpstart server.

```
#!/bin/ksh
#
#
FILE=$1
APP=$2
MODE=640
DIR=`dirname $FILE`
LOG=`basename $FILE`
OLDDIR=$DIR/old

if [ ! -d $DIR ]; then
        echo "$DIR: invalid"
        exit 255
fi

# If $OLDDIR doesn't exist, create it. If there is a problem creating it,
# just store the old logs in the current directory

if [ ! -d $OLDDIR ]; then
        mkdir $OLDDIR
        if [ $? -ne 0 ]; then
                echo "WARNING: Couldn't create $OLDDIR"
                OLDDIR=$DIR
        fi
fi

test -f $OLDDIR/$LOG.2 && mv $OLDDIR/$LOG.2 $OLDDIR/$LOG.3
test -f $OLDDIR/$LOG.1 && mv $OLDDIR/$LOG.1 $OLDDIR/$LOG.2
test -f $OLDDIR/$LOG.0 && mv $OLDDIR/$LOG.0 $OLDDIR/$LOG.1
test -f $DIR/$LOG      && mv $DIR/$LOG      $OLDDIR/$LOG.0

cp /dev/null $DIR/$LOG
chmod $MODE $DIR/$LOG

/etc/init.d/$APP stop
/etc/init.d/$APP start
```

The script is based on /usr/lib/newsyslog and from SANS' Linux/Solaris Practicum.[19] The script accepts two arguments, the log file to rotate, and which startup script to run. It rotates the current log files to the old directory and rotates the older log files. Instead of signaling with –HUP, the daemons are stopped then started. This is because syslog is prone to crash when a HUP is sent to the process and other daemons may not write to the new log file until after the daemons have been restarted.

**System Accounting**

If this system is used solely as a jumpstart server, for the most part it will not be taxed by any means - unless you are attempting to jumpstart 10 machines at the same time while compiling software. However it still doesn't hurt to enable system accounting. This information can be used to produce pretty little graphs, which are used for performance tuning, capacity planning, and detecting intrusions.[20] If an irregular spike is noticed on one of the graphs, something interesting might be happening. Also, if a system's cpu usage is slowly climbing from week to week, it might be time to talk to the boss about upgrading to the next higher system model.

The system accounting packages were installed earlier. To enable, uncomment the appropriate lines in /etc/init.d/perf and /var/spool/cron/crontabs/sys. Now run /etc/init.d/perf and system accounting will be running. The system reports are stored in /var/adm/sa and can be used viewed with **sar –f**. The raw data can be used with GNUplot, Excel, etc. anything that can graph and make nice little charts that make managers happy. The utility **sag** can be used to provide graphical output on certain terminal types. The sar data is kept around for at most a month by default. This should be changed and can be easily implemented by modifying /etc/init.d/perf and changing the **`date +%d`** calls to **`date +%Y%m%d`**. Note that **sar** will still look for files with the old naming convention and that the –f option will be needed. Create an alias to do this automatically. Also the /usr/lib/sa/sa2 script purges raw data files over a week old, comment out the find or modify to keep the files even longer. [21]

**Process Accounting**

Process accounting can be very useful to see what the users are doing as it logs every process each user has run. Each process is logged by the kernel and takes 40 bytes of data. However, the process is not logged until it is finished executing, so daemons may not be logged until system shutdown. See the **accton** manual page for information on how to analyze the data. Note that there are some dangers to running process accounting. It can cause a significant decrease in performance of about 10-20%. Also if a large number of processes are starting and finishing constantly, then it may cause problems with disk performance and it may fill the disk up. If you wish to have process accounting at system startup, add **/usr/lib/acct/accton /var/adm/pacct** to /etc/init.d/perf.[22]

**Kernel Auditing**

Kernel auditing can log just about any system call made on the system. Information about the user, which system call, what files, etc can be gathered. Kernel auditing can produce a large amount of uninteresting data. If and when a security incident occurs, this data can be extremely helpful. To enable kernel auditing, run **/etc/security/bsmconv**, however note that auditing won't start until the system is rebooted. Edit the /etc/security/audit_control file:

```
dir:/var/audit
flags:lo,ad,-all,^-fm
```

```
naflags:lo,ad
minfree:20
```

From the configuration file, the logs will be stored in /var/audit and when /var is 80% full, the system administrator will start to receive warnings. The audit daemon will continue to log to the same file until told via **audit –n**. The logs in /var/audit are in binary format. Use **praudit** to produce text dumps. Then use Perl or C to parse the data. The program **auditreduce** can be used to parse audit events based on parameters. Be sure to compress the logs on a regular basis. Also, migrate older logs to another system or tape.[23]

**Miscellany**

For some reason, if a warning message is not displayed, the judicial system thinks it is okay for other people to trespass. So because of this, a warning must be displayed whenever a user logs on. To do this, edit /etc/motd and add something along the lines of:

```
This computer system (including all hardware, software, and peripheral equipment) is the
property of YOUR COMPANY. Use of this computer system is restricted to official
YOUR COMPANY business. YOUR COMPANY reserves the right to monitor use of the
computer system at any time. Use of this computer system constitutes consent to such
monitoring. Any unauthorized access, use, or modification of the computer system can
result in disciplinary action and/or possible civil or criminal penalties.
```

Be sure to consult the legal group before adding a warning message.

Next configure the files in /etc/default[24]:

**inetinit**: by setting **TCP_STRONG_ISS=2** forces the system to use a better algorithm for randomizing TCP sequence numbers.
**cron**: set **CRONLOG=YES** to log several lines of output for each job that is executed by cron. Also set **PATH=/usr/bin:/usr/local/bin** and **SUPATH=/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin**, path is for all users and supath is for root.
**su**: Again set **PATH** and **SUPATH**. Use the same values as in cron.
**passwd**: Turn on password aging by setting **MAXWEEKS**. **PASSLENGTH** can also be adjusted, however this can actually make it easier for someone to crack a password by reducing the number of possible passwords.
**kbd**: This prevents sending the system a break signal and putting it at the PROM level. For systems in a data center, this prevents administrators from halting systems that are frozen and syncing the disks, and is usuall not necessary in a locked facility. On desktops in an open office, it is usually a good idea to disable this.
**init**: Set the **CMASK** to 077 if paranoid. This sets the umask on processes started by init. The default (022) is adequate for most.
**login**: **PATH** and **SUPATH** set as in cron. Set **UMASK** to 022 unless paranoid,

otherwise set it to 077. **RETRIES** is the number of failed login attempts before the login program has to be re-spawned. Setting this value between 3 and 5 is acceptable. To log every failed login attempt, set **SYSLOG_FAILED_LOGINS=0**.

The default login files used by all users can also be edited at this time also. If adequate entries are used in /etc/profile, /etc/login, etc., then most users won't bother with creating their own login files. However users can always ignore these settings.

Now it is time to secure the EEPROM on the system. To enable password protection that isn't too restrictive, use:

```
# eeprom security-mode=command
```

The security-mode can be set to none, command, or full. If full is selected, any time a system is rebooted, an operator and admin has to enter the password. Using command allows the system to be rebooted with human interaction, but prevents an attacker changing EEPROM settings thus causing a denial of service attack. Do not set the EEPROM password to the same as root. If the root password is cracked, then the cracker has the password to the EEPROM and can perform the same denial of service attack. If the password is changed or forgotten, the only way to recover is to install a new EEPROM from Sun. The number of invalid logins can be determined by using the command **eeprom security-#badloginsb**. This can be reset to zero at any time by issuing **eeprom security-#badlogins=0**. For the same reasons as having a MOTD for the OS, a warning message should also be displayed for the EEPROM. This can be set by:

```
# eeprom oem-banner="Authorized users only. \
        All access may be logged and reported."
# eeprom oem-banner\?=true
```

Another possible EEPROM setting is to allow each interface to use it's own MAC address. After enabling this, the system will need to be rebooted.[25]

```
# eeprom local-mac-address\?=true
```

In addition to any other settings used by applications, here are some recommended entries for /etc/system. The first group tells the operating system to disable executable stacks. This helps to protect against stack and buffer overflow attacks.

```
* Prevent and log stack-smashing attacks
set noexec_user_stack = 1
set noexec_user_stack_log = 1
```

This group of entries sets the maximum and standard number of file descriptors allowed per process. The last line sets the maximum number of process permitted per user.

```
* Set various parameters to more reasonable values
set rlim_fd_max = 1024
set rlim_fd_cur = 256
set maxuprc = 150
```

This entry prevents the system from dumping core files. This sets the maximum file size to 0 for a core dump. This prevents any attacker from using a core file as a DOS attack or using a core file to steal certain information like the /etc/shadow file.

```
* No core dump files
set sys:coredumpsize = 0
```

This entry states that any NFS client requests must originate from low number ports. On UNIX systems users must have root privileges to use any ports lower than 1024. Thus preventing any non-root user from connecting to the NFS services. This is not the case on Windows however. Any user can use any port.

```
* NFS clients to user privilege ports
set nfssrv:nfs_portmon = 1
```

The final entry for the system file is to force the hme interfaces to use 100MB Full Duplex. This entry is in place because Solaris can have problems auto negotiating with a switch. It also doesn't hurt to force the interface to the appropriate setting just so there is no doubt to what speed and mode the interface is running.[26]

```
* Set hme to 100M Full duplex.
set hme:hme_adv_autoneg_cap=0
set hme:hme_adv_100T4_cap=0
set hme:hme_adv_100fdx_cap=1
set hme:hme_adv_100hdx_cap=0
set hme:hme_adv_10fdx_cap=0
set hme:hme_adv_10hdx_cap=0
```

**Additional Software**

At this point, create both /opt/jumpstart/packages and /opt/jumpstart/files directories. These directories will be used to store packages and files that are installed on all systems that use this Jumpstart server. Also note, that while a client is being jumpstarted, the file systems are mounted with a leading /a. So the root partition is mounted as /a, and usr is mounted as /a/usr, etc. This information is used when creating postinstall script. If a postinstall script is running and the script wants to add a crontab entry, then the file to edit would actually be /a/var/spool/cron/crontabs/root.

**GNU's GCC**

Homepage: http://www.gnu.org
Versions: 2.95.3 and 3.0.2
Precompiled 2.9.5: http://www.sunfreeware.com

This system needs to build binaries that will be distributed to new servers and possibly to manually install the updated packages on current production systems. Because Solaris 8 runs in 64-bit and supports 32-bit applications, a compiler that supports both is needed. In this case, two compilers will be used, GCC version 2.95.3 in 32-bit mode, and GCC version 3.0.2 in 64-bit mode. Since a compiler is not yet on the system, one will have to be downloaded. From http://www.sunfreeware.com download the pre-built package of gcc-2.95.3 and install it. While doing this, download from http://www.gnu.org version 3.0.2 of gcc. Use the following steps to build version 3.0.2 of gcc in 64-bit mode. The first step creates a 32-bit version of the 3.0.2 compiler.

```
gzip –d gcc-3.0.2.tar.gz
tar xf gcc-3.0.2.tar
cd gcc-3.0.2
./configure --prefix=/tmp/gcc-32bit --enable-languages=c,c++
…output deleted…
make
…output deleted…
make install
```

Now set the path and clean the source tree. This step will build a 32-bit compiler that can generate 64-bit applications. So the step after this can be skipped, just be sure to set the prefix for this step to /usr/local/gcc-64bit or something along those lines.

```
PATH=/tmp/gcc-32bit/bin:$PATH
LD_LIBRARY_PATH=/tmp/gcc-32bit/lib:$LD_LIBRARY_PATH
export PATH LD_LIBRARY_PATH
make clean
./configure --prefix=/tmp/gcc-64bit --enable-languages=c,c++ sparcv9-sun-solaris2
make
make install
```

Again prepend the path to add the new binaries and clean the source tree. This last step will create a 64-bit version of gcc.

```
PATH=/tmp/gcc-64bit/bin:$PATH
LD_LIBRARY_PATH=/tmp/gcc-64bit/lib:$LD_LIBRARY_PATH
export PATH LD_LIBRARY_PATH
make clean
./configure --prefix =/usr/local/gcc-64bit --enable-languages=c sparcv9-sun-solaris2
make
make install
```

Now 64-bit executables can be made. Be sure to keep around a 32-bit compiler because not all source code plays nicely with the 64-bit compiler. One technique that can make life easier is to create a file that has environment settings with the 64-bit compiler binaries and libraries at the front of the path and just source the file when needed.[27]

**TCP Wrappers**

TCP Wrappers allows the system administrators to limit who can connect to each system and what protocol can be used. An example of this would be to allow the administrators on network 10.1.1.0 access to ssh on a web server, while denying everyone else. While this doesn't prevent the system from listening on the SSH port, it will prevent access from an unauthorized networks while logging any failed attempts. TCP wrappers can be installed via packages, but it is not necessary for jumpstart. As long as OpenSSH is built with TCP Wrappers support, TCP Wrappers won't need to be on each system that is jumpstarted.

Even though this system isn't using IPv6, be sure to download the version that supports it. This way if IPv6 is enabled, this system will be ready. Before compiling the software, edit the make file and set REAL_DAEMON_DIR=/usr/sbin and set FACILITY=LOG_AUTH. If using Solaris 8 package utilities, add the following lines, otherwise just run **make install** and continue on to the configuration files.

```
DESTDIR         = ./pkg
BASEDIR         = /usr/local
INSTALL         = /usr/ucb/install
DESTSBIN        = $(BASEDIR)/sbin
DESTMAN         = $(BASEDIR)/man
DESTLIB         = $(BASEDIR)/lib

OTHERDIRS       = $(DESTDIR)$(DESTSBIN) \
            $(DESTDIR)$(DESTMAN) \
            $(DESTDIR)$(DESTMAN)/man3 \
            $(DESTDIR)$(DESTMAN)/man5 \
            $(DESTDIR)$(DESTMAN)/man8 \
            $(DESTDIR)$(DESTLIB)

install:  $(DESTDIR)$(BASEDIR) $(OTHERFILES) $(OTHERDIRS) sunos5
      ${INSTALL} -c -m 755 tcpd ${DESTDIR}${DESTSBIN} ; \
      ${INSTALL} -c -m 755 tcpdmatch ${DESTDIR}${DESTSBIN} ; \
      ${INSTALL} -c -m 755 tcpdchk ${DESTDIR}${DESTSBIN} ; \
      ${INSTALL} -c -m 755 safe_finger ${DESTDIR}${DESTSBIN} ; \
      ${INSTALL} -c -m 755 try-from ${DESTDIR}${DESTSBIN} ; \
      ${INSTALL} -c -m 444 tcpd.h ${DESTDIR}${DESTLIB} ; \
      ${INSTALL} -c -m 444 libwrap.a ${DESTDIR}${DESTLIB} ; \
      ${INSTALL} -c -m 444 -o bin -g bin hosts_access.3
${DESTDIR}${DESTMAN}/man3/hosts_access.3 ; \
      ${INSTALL} -c -m 444 -o bin -g bin hosts_access.5
${DESTDIR}${DESTMAN}/man5/hosts_access.5 ; \
      ${INSTALL} -c -m 444 -o bin -g bin hosts_options.5
${DESTDIR}${DESTMAN}/man5/hosts_options.5 ; \
      ${INSTALL} -c -m 444 -o bin -g bin tcpd.8
${DESTDIR}${DESTMAN}/man8/tcpd.8

$(DESTDIR)${BASEDIR} ${OTHERDIRS}:
      mkdir -p $@; \
```

Also edit the "**clean:**" to look like:

```
clean:
      rm -f tcpd miscd safe_finger tcpdmatch tcpdchk try-from *.[oa] core
      rm -rf ./pkg \
      cflags
```

Now just run **make install**. After this has completed, use the Solaris 8 package tools to create a TCP Wrappers package and install it. See http://www.sunfreeware.com/pkgadd.html on how to create packages. Since this package will only be installed on this system, there is no need to copy it to /opt/jumpstart/packages.

To configure, edit /etc/hosts.allow and add entries to allow only certain networks to ssh in.

```
sshd: 10.1.1.0/255.255.255.0
```

Then edit /etc/hosts.deny and add the following:

```
ALL: ALL: /usr/bin/mailx \
        -s “%s: connection attempt from %c” \
        root@mydomain.com
```

Change the ownership to root:root and change permissions to 600 for both hosts.deny and hosts.allow. These files allow ssh connectivity from the 10.1.1.0/24 network and deny all other ssh access to the system. If an attempt is made from any other network than 10.1.1.0/24, then an email will be sent to alert the administrator of unauthorized access attempts.[28]

**OpenSSL**

Get the OpenSSL source from http://www.openssl.org/source/. Uncompress and untar the file. Again packages do not have to be used, if they are being used run:

```
# config --prefix=./pkg/usr/local --openssldir=./pkg/usr/local/openssl.
```

Otherwise just run **config**. Then run **make; make install**. After OpenSSL has compiled, create a package using the same steps as TCP Wrappers and install the OpenSSL package.

**OpenSSH**

The source for the latest OpenSSH can be found at http://www.openssh.com/portable.html. Again uncompress and untar the file. Be sure to build OpenSSH with TCP Wrapper support. Run:

```
#configure --with-tcp-wrappers=/usr/local/lib \
>         --with-default-path=”/bin:/usr/bin:/usr/local/bin”
# make
# make DESTDIR=/tmp/pkg
# mkdir –p /tmp/pkg/etc/init.d
# mkdir –p /tmp/pkg/etc/rc2.d
```

Create a script that starts OpenSSH upon system startup and save it to /tmp/pkg/etc/init.d. Then create a link in /tmp/pkg/etc/rc2.d to the startup script. Make sure it is owned by root:root and the permissions are 744.

```
#!/bin/sh
# start/stop openssh

BASEDIR=/usr/local

case $1 in
	start)
		# use the background because the EGD can sometimes take some time..
		$BASEDIR/sbin/sshd &
		;;
	stop)
		# nice kill, sleep 5 seconds, then brute force the kill
		pkill sshd
		sleep 5
		pkill -9 sshd
		;;
	*)
		echo "$0 usage: $0 [start|stop]"
		;;
esac
```

On the next page is a copy of a basic secure /tmp/pkg/usr/local/etc/sshd_config. All authentication using rhosts is disabled for security purposes. Logging is done to LOG_AUTH via syslog to track everything. Authentication can be done using either RSA or DSA keys, or using passwords, however empty passwords are not acceptable. Also, root is unable to login via ssh. This may prevent the ability to do some system administration tasks root, but this option can be changed to **without-password**. That will allow root to login as long as DSA or RSA keys have been distributed appropriately. Also note the MOTD is not being printed by sshd, this is because it will already be done via login. Check to make sure that this configuration file is set to read only and owned by root:root.[29]

```
Port 22
Protocol 2,1
ListenAddress 0.0.0.0

# HostKeys
HostKey /usr/local/etc/ssh_host_key
HostKey /usr/local/etc/ssh_host_rsa_key
HostKey /usr/local/etc/ssh_host_dsa_key

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 900
ServerKeyBits 1024

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 180
PermitRootLogin no
StrictModes yes
RSAAuthentication yes
DSAAuthentication yes
PubkeyAuthentication yes

# rhosts authentication should not be used
RhostsAuthentication no
IgnoreRhosts yes
RhostsRSAAuthentication no
HostbasedAuthentication no
IgnoreUserKnownHosts yes

PasswordAuthentication yes
PermitEmptyPasswords no

X11Forwarding yes
X11DisplayOffset 10
PrintMotd no
KeepAlive no
UseLogin no

Subsystem       sftp    /usr/local/libexec/sftp-server
```

In the root package directory /tmp/pkg, create the postinstall script for the package.

```
#!/sbin/sh

if [ ! -d /a ];
then
        sysconfdir=/usr/local/etc
        bindir=/usr/local/bin
        sbindir=/usr/local/sbin

        if [ -f "$sysconfdir/ssh_host_key" ] ; then
                echo "$sysconfdir/ssh_host_key already exists, skipping."
        else
                $bindir/ssh-keygen -t rsa1 -f $sysconfdir/ssh_host_key -N ""
        fi
        if [ -f $sysconfdir/ssh_host_dsa_key ] ; then
                echo "$sysconfdir/ssh_host_dsa_key already exists, skipping."
        else
                $bindir/ssh-keygen -t dsa -f $sysconfdir/ssh_host_dsa_key -N ""
        fi
        if [ -f $sysconfdir/ssh_host_rsa_key ] ; then
                echo "$sysconfdir/ssh_host_rsa_key already exists, skipping."
        else
                $bindir/ssh-keygen -t rsa -f $sysconfdir/ssh_host_rsa_key -N ""
        fi
fi
```

This checks to see if it is being installed during a jumpstart, if so exit. Otherwise, create the host keys if they don't already exist. The reason the keys are created during jumpstart is because the entropy gathering routine uses /usr/local/etc/ssh_prng_cmds and not all commands are available at the time of jumpstart. Plus the entropy gathering is done using commands run against log files. During a jumpstart the log files the new server are more than likely going to be small if containing any data at all. Thus creating very little entropy.

**Other Needed Software**

Flex Homepage: http://www.gnu.org
Bison Homepage: http://www.gnu.org
libpcap: http://www.tcpdump.org

These tools are needed to compile other software. To compile, use the 32-bit compiler for all sources and just go through the basic steps for all.

```
# ./configure
# make
# make install
```

Packages shouldn't be needed because the software will only be needed on the Jumpstart server.

**LSOF**

Download: ftp://vic.cc.purdue.edu/pub/tools/unix/lsof
Version: 4.60

LSOF is a wonderful utility, which is used to list open files. It can be used to know which processes are using what files, directories, port, and/or network connections. Sun has been known to turn on services after patching that were disabled prior to patching and lsof is perfect to see what Sun has been kind enough to turn back on. Here is an example on how to check what ports are listening:

```
# lsof -i | grep LISTEN
rpcbind    146 root    6u  IPv4 0x30001482e28      0t0  TCP *:sunrpc (LISTEN)
sshd       238 root    5u  IPv4 0x300015fd470      0t0  TCP *:22 (LISTEN)
mountd     266 root    8u  IPv4 0x300015fccf0      0t0  TCP *:32771 (LISTEN)
nfsd       268 root    5u  IPv4 0x300015fc570      0t0  TCP *:nfsd (LISTEN)
rpc.bootp  273 root    1u  IPv4 0x300017afe78      0t0  TCP *:32772 (LISTEN)
```

To install lsof, download and unpack it. Since Solaris 8 is running in 64-bit mode, lsof needs to be compiled with the 64-bit compiler. The **Configure** script prompts for some options that need to be set. The options that are set allow only root to use this program. This is done because a lot of useful information can be gotten if an attacker was able to access a system and use this utility. Another option is that the executable will be run on other systems. Since every system won't have a compiler on it, the software will have to have this option enabled.

```
PATH=/usr/local/gcc-64bit/bin:$PATH
LD_LIBRARY_PATH=/usr/local/gcc-64bit/lib:$LD_LIBRARY_PATH
export PATH LD_LIBRARY_PATH
# ./Configure solaris
… skip output
Do you want to take inventory (y|n) [y]? y
...
Do you want to customize (y|n) [y]? y
...
When HASSECURITY is disabled, anyone may use lsof to examine all
open files.

HASSECURITY is disabled.

Enable HASSECURITY (y|n) [n]? y
…
Disable HASDCACHE (y|n) [n]? y
...
Disable HASKERNIDCK (y|n) [n]? y
...
Do you want to rename machine.h to machine.h.old and replace it with
new_machine.h (y|n) [y]? y
…
# make
```

Since this software will need to be loaded on all clients that are jumpstarted, a package can be made for this program. Although since it is only two files, that is not necessary. Just put the two files in /opt/jumpstart/files and copy them over via the file copy script. See the filecp.sh script in the Jumpstart Software Configuration section for examples.

```
# mkdir –p /tmp/pkg/usr/local/bin /tmp/usr/local/man/man8
# install –c /tmp/pkg/usr/local/bin –m 2755 –f sys lsof
# install –c /tmp/pkg/usr/local/man/man8 –m 444 lsof.8
```

Using the package utilities create a package, install it, and copy the package to the /opt/jumpstart/packages directory.

**Sudo**

Homepage: http://www.courtesan.com/sudo/
Version: 1.6.3p7

This program allows users to run specific commands as root. These commands are set by the system administrator. Users are challenged for their password when these commands are used in conjunction with sudo. A log entry is made when sudo is ran, thus creating an audit trail. This allows the system administrator to give users access to commands for which they need root access, without having to give the user the root password. An example of this is night operators that run backups.

This utility can be built in 64-bit mode, but there really is nothing to gain by doing this. To compile the software:

```
# ./configure
# make
# make prefix=/tmp/sudo/usr/local install
```

Change directories to /tmp/sudo and make a package. Consider creating a default sudoers file in /tmp/sudo/etc/sudoers with the following entries.

```
# User privilege specification
root    ALL=(ALL) ALL
%sysadmin       ALL=(ALL) ALL
```

This allows any user in the sysadmin group to run any command via sudo. If this is done, be sure to put only system administrators in the sysadmin group. If this file is left blank, no one will be able to use sudo.

**IP Filter**

Homepage: http://coombs.anu.edu.au/~avalon/
Version: 3.4.20
Note: At this time the current version is 3.4.22, however there are problems with IP Filter that causes the system to lock up without leaving any logs as to why the system crashed.

IP Filter is freeware packet filtering software. It can turn any UNIX system into a firewall. In this case, however it will be used to limit access to the system. The writers of IP filter were kind enough to include in the Solaris Makefile an option to make packages. To compile the software be sure to use Solaris's **make** and not GNU's. ( Put /usr/ccs/bin at front of path ) Also be sure to use the 64-bit compiler to compile the software. Prior to running the configure script, the **buildsunos** script must be edited. Change the line **XARCH32=""** to **XARCH32="-m32"**. After that start the compiling.

```
# make solaris
# cd SunOS5
# make package
```

The last command will create the packages and install them. After the packages have installed, change to the source/SunOS5/sparc-8 directory and copy ipf.pkg to /opt/jumpstart/packages. The installation of this software will require a reboot. Do not reboot the system at this time. Create the initial rules file **/etc/opt/ipf/ipf.conf**. For this system, ssh will be allowed from the local network and the services required for jumpstart will be permitted only on the second interface. This configuration file allows on traffic on the loopback interface, blocks fragmented packets and packets with IP options set, allows ssh from the local network on the primary interface, allows all outbound traffic, and blocks all other inbound traffic. There are two rules in here for jumpstart at all times only. These are used for arp and were difficult to just add and remove on the fly as the rest of the rules for jumpstart are done. (See the jumpstart section for these additional rules.)

```
# Allow all loopback traffic
pass in quick on lo0 all
pass out quick on lo0 all
#
# Block fragmented and packets with IP options
block in log quick all with short
block in log quick all with ipopts
#
# Allow ssh
pass in quick on hme0 proto tcp from 10.1.1.0/24 to loki/32 port = 22
#
# For jumpstart
pass in quick on hme1 from 192.168.200.0/24 to 255.255.255.255/32
pass in quick on hme1 from 192.168.200.0/24 to 192.168.200.255/32

# Clean up rule
block in log all
#
pass out quick proto tcp/udp all keep state
pass out quick proto icmp all keep state
```

NOTE: Do not reboot the server until either all software in this section is installed. This is because /usr will be mounted as read only and no other software will be able to install.

**Logcheck**

Homepage: http://www.psionic.com/abacus/logcheck
Version: 1.1.1

Logcheck checks the log files and emails alerts to root whenever it notices something out of the ordinary. If these systems are going to be used in a large environment, administrators may want to consider logging to a syslog server and running logcheck and/or custom scripts there. To build logcheck, first modify the Makefile and set the install directories to precede with /tmp/logcheck, set the tmp directory to /tmp/logcheck/var/log/logcheck, and change the installation directory for logcheck.sh to the bin directory instead of etc.

```
# mkdir –p /tmp/logcheck/var/log/logcheck /tmp/logcheck/usr/local/bin \
>         /tmp/logcheck/usr/local/etc
# make sun
```

After the software has compiled and been installed, the configuration files need to be edited. Add all the log files to parse, add another keyword to trigger on, and other configuration issues. First edit /tmp/logcheck/usr/local/etc/hacking.violations and add **PORTSCAN**. This entry configures logcheck to also trigger on any port-scans in the snort logs. Next edit /tmp/logcheck/usr/local/bin/logcheck.sh and change TMPDIR=/var/log/logcheck, comment out **MAIL=mail** and uncomment **MAIL=mailx** and add:

```
$LOGTAIL /var/log/syslog > $TMPDIR/check.$$
$LOGTAIL /var/log/authlog >> $TMPDIR/check.$$
$LOGTAIL /var/log/snort/alert >> $TMPDIR/check.$$
$LOGTAIL /var/adm/messages >> $TMPDIR/check.$$
```

Before the package is made, create a postinstall that creates a cron entry that runs logcheck every hour. If the package is being added during a jumpstart, it will still add a cron entry.

```
if [ -d /a ];
then
      basedir=/a
else
      basedir
fi
echo "# hourly log check" >> $basedir/var/spool/cron/crontabs/root
echo "0 * * * * /usr/local/etc/logcheck.sh 2>&1" >>$basedir/var/spool/cron/crontabs/root
```

Again, create a package, install it, and test it prior to copying the package to /opt/jumpstart/packages. If satisfied with the current output, then copy it to /opt/jumpstart/packages. Otherwise, modify the current installed configuration files to suit and copy over the configuration files in the /tmp/logcheck directory structure.

**Tripwire**

Homepage: http://www.tripwire.com/products/tripwire_asr
Version: 1.3.1-1

Note: This is the Academic Source and per the End User License Agreement, can only be installed on one system in a corporation.

From Tripwire documentation:

"Tripwire is a file integrity assessment tool, a utility that compares a designated set of files and directories against information stored in a previously generated database."[30] Basically Tripwire monitors specified files and directories for any changes. If there are any, the administrator needs to determine whether or not it is due to an attack. If not, update the database and continue normal operations, otherwise take the system off the network and start doing the analysis.

To compile Tripwire, use the 32-bit compiler. (64-bit will still compile, however after the compilation the tests will all fail.) Edit include/config.h and make sure the include is **#include "../configs/conf-svr4.h"**. Next edit the Makefile and modify/create the following entries:

```
DESTDIR = /tmp/tripwire/usr/local/bin/tw
DATADIR = /tmp/tripwire/var/tripwire
MANDIR  = /tmp/tripwire/usr/local/man
MANDIR5  = /tmp/tripwire/usr/local/man/man5
MANDIR8  = /tmp/tripwire/usr/local/man/man8
LEX     = flex
YACC    = bison -y
```

Now run:

```
# make
# make test
# make install
# mkdir –p /tmp/tripwire/etc/opt/tw
```

The last step is for the Tripwire configuration. The configuration file for the jumpstart server is:

```
# text omitted
#  First, root's "home"
=/                          L
/.rhosts                    R      # may not exist
/.profile                   R      # may not exist
/.cshrc                     R      # may not exist
/.login                     R      # may not exist
#/.exrc                     R      # may not exist
/.logout                    R      # may not exist
#/.emacs                    R      # may not exist
/.forward                   R      # may not exist
/.netrc                     R      # may not exist
#/.mailrc                   R      # may not exist
/.ssh/authorized_keys       R
/.ssh/authorized_keys2      R
/.ssh/identity.pub          R
/.ssh/identity              R
/.ssh/id_dsa.pub            R
/.ssh/id_dsa                R

# Unix itself
/kernel                     R
/usr/kernel                 R

# Device files
/dev                        L
/devices                    L
=/devices/pseudo            L

# Configuration files
/etc                        R
/etc/dumpdates              L
/etc/passwd                 L
=/etc/saf                   L
/etc/shadow                 L
/etc/ttydefs                L
!/etc/mnttab                R
!/etc/sharetab              R
/usr/local/etc              R
/etc/hosts                  E+pug
/etc/ethers                 E+pug
/etc/bootparams             E+pug
/etc/utmppipe               E+pug
```

```
# System directories
=/tmp                           L
=/var/tmp                       L
=/proc                          L

/usr                            R-2

# Critical binary directories
/sbin                           R
/usr/sbin                       R
/usr/local/bin                  R
/usr/local/sbin                 R
/usr/ccs/bin                    R
/usr/ucb/bin                    R

# Libraries
/usr/lib                        R
/usr/local/lib                  R

/ufsboot                        R

# Var file system
=/var                           L
=/var/adm                       L
/var/adm/utmp                           L
/var/adm/wtmp                   L
/var/adm/wtmpx                  L
/var/adm/sulog                          L
=/var/adm/sa                    L
=/var/spool                     L
/var/spool/cron/crontabs        R

# /opt file system
/opt/                           R
```

Please see the Tripwire manual for a description of the configuration file. With this configuration file, most of the system configuration files in /etc, dot files in root's home, binary directories in /usr, library directories in /usr, kernel files, IP filter, crontabs, and the jumpstart files are checked with two checksums, MD5 and Snefru. The rest of /usr is checked with just MD5. The log files are also checked, however Tripwire doesn't check the file size and last access time. Notice that some files in /etc are using "E+pug", this is because the jumpstart scripts modifies those files every time a new client system is jumpstarted. Those settings check to make sure the permissions, owner and group have not changed. The basic configuration file that is installed via jumpstart is pretty much the same as above, except for the files in /etc/ with “E+pug” entries above are not in the file, those file are still handled by the entry for /etc. On all other systems, those files should rarely ever change.

The postinstall script here will create a generic database after the package has installed. If the package is being installed during a jumpstart, then the script will create another startup script that creates a new tripwire database. The postinstall script will also create a crontab entry emailing the desired person a tripwire report.

```
#!/bin/sh
# If jumpstarting, then at first boot create the database, otherwise create the
# database now
if [ -d /a ];
then
        echo "cd /var/tripwire" > /a/etc/rc3.d/S99tw
        echo "/usr/local/bin/tw/tripwire -initialize -c /etc/opt/tw/tw.config" \
                >> /a/etc/rc3.d/S99tw
        echo "mv databases/tw.db_$myhost /var/tripwire" >> /a/etc/rc3.d/S99tw
        echo "rmdir databases" >> /a/etc/rc3.d/S99tw
        echo "rm /etc/rc3.d/S99tw" >> /a/etc/rc3.d/S99tw
        echo "#"
        echo "# Tripwire check" >> /a/var/spool/cron/crontabs/root
        echo "0 10 * * * /usr/local/bin/tw/tripwire -c /etc/opt/tw/tw.config \
                2>&1 | /usr/bin/mailx -s 'Tripwire output from `uname -n`' \
                someone@here.com" >> /a/var/spool/cron/crontabs/root
        echo "cat << EOF" >> /a/etc/rc3.d/S99tw
        echo "---------------------------------------------------------------" >> \
                /a/etc/rc3.d/S99tw
        echo "A generic tripwire database has been created for this system." >> \
                /a/etc/rc3.d/S99tw
        echo "It is highly recommended that you customize /etc/opt/tw/tw.config" >> \
                /a/etc/rc3.d/S99tw
        echo "for this system and rerun:" >> /a/etc/rc3.d/S99tw
        echo "/usr/local/bin/tw/tripwire -initialize -c /etc/opt/tw.config" >> \
        /a/etc/rc3.d/S99tw
        echo "and put the new database in /var/tripwire" >> /a/etc/rc3.d/S99tw
        echo "---------------------------------------------------------------" >> \
                /a/etc/rc3.d/S99tw
        echo "EOF" >> /a/etc/rc3.d/S99tw
else
        cd /var/tripwire
        /usr/local/bin/tw/tripwire -initialize -c /etc/opt/tw/tw.config
        mv databases/tw.db_$myhost /var/tripwire
        rmdir databases
        echo "#"
        echo "# Tripwire check" >> /var/spool/cron/crontabs/root
        echo "0 10 * * * /usr/local/bin/tw/tripwire -c /etc/opt/tw/tw.config 2>&1 | \
        /usr/bin/mailx -s 'Tripwire output from `uname -n`' \
        someone@here.com" >> /var/spool/cron/crontabs/root
        echo "----------------------------------------------------------------"
        echo "A generic tripwire database has been created for this system."
        echo "It is highly recommended that you customize /etc/opt/tw/tw.config"
        echo "for this system and rerun:"
        echo "/usr/local/bin/tw/tripwire -initialize -c /etc/opt/tw.config"
        echo "and put the new database in /var/tripwire"
        echo "----------------------------------------------------------------"
fi
```

**Snort**
Homepage: http://www.snort.org
Version: 1.8.3

Snort is a multipurpose utility. First, it can be used to do network dumps. It can be more useful than snoop, the network sniffer that is installed with Solaris, because it can display a portion of the network traffic in real time and in a more compact form. Below is an example of a packet being display by snort. The text was shrunk and set to a more readable font for this example. Notice that the traffic is on port 22, which is used by ssh. So the packet contents on the right are not readable. If a protocol that doesn't use encryption is used, then the data in the packets can be viewed. Such as login names and passwords.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

12/22-02:24:53.991070 1.2.3.5:22 -> 1.2.3.4:1718
TCP TTL:64 TOS:0x10 ID:33849 IpLen:20 DgmLen:156 DF
***AP*** Seq: 0xDFEAF24E  Ack: 0x3D8F3  Win: 0x60F4  TcpLen: 20
00 00 00 6E 0E 4D 59 E2 38 AC 40 5E E4 63 24 9D  ...n.MY.8.@^.c$.
99 B0 F5 D5 94 24 B3 8C 79 D9 DE 66 80 97 4F 53  .....$..y..f..OS
60 F0 2C 67 69 04 DC 69 27 64 0E C6 20 F8 66 EE  `.,gi..i'd.. .f.
8C C8 C3 67 B2 8D FE 2B C5 DD D8 47 CF B7 60 1B  ...g...+...G..`.
FA FD D2 09 73 36 CD 28 39 B2 6F C1 69 C7 F8 F6  ....s6.(9.o.i...
56 FD C3 48 82 8B F0 33 86 C3 1D 1D DB E1 5C 06  V..H...3......\.
0E A3 9B FD E6 CE 69 E0 6E 4C B5 20 2B 4E 1F E3  ......i.nL. +N..
5F A4 38 F8                                      _.8.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
```

The other way to run snort, is in intrusion detection mode. By running snort in this mode, administrators can tell when a system is being port-scanned or is under a number of other attacks.

To compile snort, use a 32-bit compiler and for some reason, the configure script would not complete and would fail on the libpcap detection. Even when the **--with-**

**includes** and **--with-libraries** were set to the appropriate path it would not configure. If this occurs when **configure** is being run, try logging off and logging back in and that should fix the issue.

```
# configure --prefix=/tmp/snort
# make; make install
# mkdir –p /tmp/snort/etc/opt/snort
```

The last line creates a directory where the snort rules are stored. Copy the snortrules.tar.gz there and unpack them. Consider creating a default configuration file in the snort.conf for distributing to all systems. A postinstall is needed that will install in snort files in the appropriate directories if this is during a jumpstart, by default this script will add the systems current network as its home network:

```
#!/bin/sh
#
#
echo "------------------------------------------"
echo "Note:"
echo ""
echo "If you add more interfaces, please edit"
echo "/etc/opt/snort/snort.conf and add entries"
echo "to the HOME_NET variable"
echo "------------------------------------------"

if [ -d /a ];
then
      basedir=/a
else
      basedir=
fi

HOSTNAME=`/usr/bin/hostname`
echo "var HOME_NET [$HOSTNAME/24]" >$basedir/tmp/snort.conf.new
cat $basedir/etc/opt/snort/snort.conf >> $basedir/tmp/snort.conf.new
mv $basedir/etc/opt/snort/snort.conf $basedir/etc/opt/snort/snort.conf.orig
mv $basedir/tmp/snort.conf.new $basedir/etc/opt/snort/snort.conf
```

The startup script in /tmp/snort/etc/rc2.d/S99snort should be linked to ../init.d/snort. This script will only work on systems with qfe interfaces or hme interfaces. It will startup a process on each interface that matches.

```
#!/bin/sh
# Startup script to run snort in intrusion detection mode
PATH=/sbin:/usr/bin:/usr/sbin:/usr/local/bin
CONFIG=/etc/opt/snort/snort.conf
SNORT=/usr/local/bin/snort
HOSTNAME=`hostname`
export PATH CONFIG SNORT HOSTNAME
case "$1" in
	'start')
		if [ -f $CONFIG ];
		then
			ifaces=`ifconfig -a | egrep -i 'hme|qfe' | awk -F: ' { print $1  } '`
			for i in $ifaces
			do
				echo "Starting snort on $i"
				$SNORT -i $i -d -o -D -c $CONFIG
				if [ $? -ne 0 ];
				then
					echo "Error starting snort on interface $i" >&2
					exit 255
				fi
			done
		else
			echo "$CONFIG doesn't exist, can't start snort" >&2
		fi
		;;
	'stop')
		# Stop snort
		if [ -x /usr/sbin/pkill ]; then
			pkill snort
		else
			ifaces=`ifconfig -a | egrep -i 'hme|qfe' | awk -F: ' { print $1 } '`
			for i in $ifaces
			do
				if [ -f /var/run/snort_$i.pid ];
				then
					kill `cat /var/run/snort_$i.pid`
				fi
			done
		fi
		;;
	*)	 # usage
		echo "usage: $0 start|stop" >&2
		exit 1
```

```
;;
esac
```

The snort logs will be parsed by the logcheck utility and the logs are stored in /var/log/snort.

**RPCBIND**

Homepage: http://www.porcupine.org
Version: 2.1

Since this system will be running rpcbind, a more secure version is desired.[31] This port of rpcbind provides interaction with TCP Wrappers. To compile this source, set WRAP_DIR to location of the TCP Wrapper library and run **make CC=gcc**. Since this is the only system using this software, copy the new binary to /usr/local/sbin and move the old to /usr/sbin/rpcbind.old. Next modify the startup script by copying /etc/init.d/rcp to /etc/init.d/newrpc and change all references to the new rpcbind in /usr/local/sbin. Edit the /etc/hosts.allow to allow the subnet of the second interface of the jumpstart server and add the following entries:

```
rpcbind: your.sub.net.number/your.sub.net.mask
rpcbind: 255.255.255.255 0.0.0.0
```

The first entry is to allow system on the same network access to the jumpstart server. The other entry is to allow broadcast information which is also required for the jumpstart and rpcbind. The hosts.deny already has an entry for everything else, so it should not need to be edited.

**TARA**

Homepage: http://www-arc.com/tara/index.html
Version: 2.0.9

This software provides host-based security scanning. There are no binaries to be compiled because it is all scripts, but a Makefile is provided, although not all files are copied for some reason.[32] Creating a TARA package requires a little extra work. To install, edit the make file and set:

```
TIGERHOME=/usr/local/tiger
TIGERWORK=/var/tiger/work
TIGERLOGS=/var/tiger/logs
TIGERBIN=/usr/local/tiger/bin
```

Next copy scripts/check_* to /usr/local/tiger/scripts. Now create the directory structure for creating a TARA package:

```
# mkdir –p /tmp/tiger/usr/local /tmp/tiger/var/tiger/work /tmp/tiger/var/tiger/logs
# mv /usr/local/tiger /tmp/tiger/usr/local
```

Edit /tmp/tiger/usr/local/tiger/tigerrc and set Tiger_FSScan_WDIR=Y.[33] This tells TARA to check for world writeable directories. Create a package, install, test, and then copy it to /opt/jumpstart/packages. A copy of the postinstall script is below. The script will automatically install a cronjob to run TARA once a day.

```
#!/bin/sh
if [ -d /a ];
then
        basedir=/a
else
        basedir=
fi

echo "# Tara/Tiger" >> $basedir/var/spool/cron/crontabs/root
echo "0 5 * * * /usr/local/tiger/tiger > /dev/null 2>&1" >>
$basedir/var/spool/cron/crontabs/root
```

**fix-modes**

Homepage: None
Download: **ftp://ftp.science.uva.nl/pub/solaris/fix-modes.tar.gz**
Version: 2.8

**fix-modes** is a tool, which corrects insecure file/directory permissions. It uses the **/var/sadm/install/contents** file which contains information about which packages are installed. The program also updates the information in the contents file reflecting the changes made.[34] Download and unpack the tar ball. To compile the software use:

```
# make CC=gcc
```

Now to fix permissions on this system, go ahead and run **fix-modes**. Then, create a new tar ball and copy the software over to the /opt/jumpstart/files directory.

**MD5**

Download: ftp://ftp.cerias.purdue.edu/pub/tools/unix/crypto/md5/
Version: 6-14-2000

This utility is used to validate checksums for source and other software that has been downloaded. There is only one executable so no package is required. Compile it and then copy it to /usr/local/bin and to /opt/jumpstart/files.

**NMAP**

Homepage: http://www.insecure.org/nmap/
Version: 2.54BETA22

Note: Do not install this on the Jumpstart Server or include this in the Jumpstart configuration. This software will be used in ongoing maintenance. This utility should be run from another secured system. This function of nmap is to scan for open ports. To install:

```
# ./configure
# make
# make install
```

**Final Touch Up**

**Initialize Second NIC**

As designed the system needs to have the second interface configured before a jumpstart can take place. To configure the system so that the second interface will be live at the next boot a few files need to be edited. First create the /etc/hostname.hme1 and add the appropriate hostname. Then edit /etc/hosts and /etc/hosts.orig. Add the hostname and IP Address to this file. The reason for the hosts.orig file is explained in the addclient portion of the Jumpstart configuration below.

```
# echo “loki-js” > /etc/hostname.hme1
# vi /etc/hosts
Edit file
# vi /etc/hosts.orig
Edit file
```

**Reboot**

At this point, the operating system installation and configuration is done. All the necessary software is installed to permit secure remote access. At this time, reboot the system to ensure that all configuration changes are in place.

```
# /sbin/init 6
```

## Jumpstart Configuration

To install the jumpstart software, first the directory structure is needed. For this setup /opt/jumpstart is being used as the root directory. Underneath that, create the following directories:[35]

| Directory | Description |
|---|---|
| OS | This contains the OS only. Multiple versions can be stored underneath this directory. |
| sysidcfg | System Identification files about the client are here. |
| Profiles | Configuration files for the |
| Begin | All scripts to be executed before the jumpstart begins are placed here. |
| Finish | The finish scripts that are run after the OS has been installed are stored here. |
| Mnt | This a temporary mount point for tftpboot slice, this is used be /tftpboot is mounted read only. |
| packages | Any additional third party packages that need to be installed are stored here. |
| Patches | OS patches are stored here. |
| Files | Any files that are copied to the new server are placed here. |
| Scripts | Any script that is called by the finish or begin scripts are here. |

**Solaris Operating System Image**

The next step is to install the Operating System Software, to do this insert the first Solaris Operating Environment CD (1 of 2) into the CD-ROM drive, mount the file system and install the Solaris OE image.

```
# mount –o ro –F hsfs /dev/dsk/c0t6d0s0 /mnt
# cd /mnt/Solaris_8/Tools
# ./setup_install_server /opt/jumpstart/OS
…output omitted…
```

Unmount the CD-ROM and insert the next disk and continue with the install.

```
# mount –r ro –F hsfs /dev/dsk/c0t6d0s0 /mnt
# cd /mnt/Solaris_8/Tools
# ./add_to_install_server /opt/jumpstart/OS
…output ommited…
```

Next the directory must be made available via NFS. Edit /etc/dfs/dfstab and add:

```
share –f nfs –o,ro anon=0 –d "Jumpstart Directory" /opt/jumpstart
```

Then enter **shareall** at the command line to start sharing the jumpstart directory.[36]

## Creating sysidcfg

This file contains information required to automate an installation. It contains configuration information for the new system such as locale, timezone, root password, etc.

/opt/jumpstart/sysidcfg/sysidcfg:[37]

```
system_locale=en_US
timezone=GMT
network_interface=primary {netmask=255.255.255.0 protocol_ipv6=no}
terminal=vt100
security_policy=NONE
root_passwrd=Something here
name_service=NONE
timeserver=localhost
```

The file is, for the most part, self-explanatory. The root password is an encrypted hash entry as in /etc/shadow. Do not make this root password the same as the Jumpstart server. The system administrator should change the entry for the root password prior to every jumpstart. The keyword name_service tells the client what to use after the installation completes, entries can be NIS+, NIS, LDAP, DNS, or NONE. All entries for name_service except for NONE will require additional information. This configuration will be using NONE. The option timeserver is set to localhost because there is no NTP server running on the jumpstart network. (However one could be configured on the jumpstart server if desired.)

## System Profiles

These files contain the information for the operating system install of the jumpstart process. The profiles contain keywords stating which install type, the system type, additional packages, etc. For the server profiles on this jumpstart server, the methods used to install the OS on this server are followed. The only difference is that new clients do not have the additional packages used to compile software. Other than that, the Operating System software is essentially the same. A profile for servers without the need to remotely run X based applications and with a root disk larger than 9GB is shown here, additional configurations will be shown in Appendix A.

/opt/jumpstart/profiles/server_large.profile

```
# This install is for servers without a need to run X-based apps
# This is for root disks => 9GB
#
# Install type
install_type    initial_install
system_type     standalone

# Cluster
cluster SUNWCreq

# additional packages
package SUNWadmc        add
package SUNWadmfw       add
package SUNWntpr        add
package SUNWntpu        add
package SUNWaccr        add
package SUNWaccu        add
package SUNWzlib        add
package SUNWzlibx       add
package SUNWgzip        add
package SUNWter         add

# Disk layout
root_device c0t0d0s0
partitioning explicit
filesys rootdisk.s0 265 / logging
filesys rootdisk.s1 auto swap
filesys rootdisk.s3 1024 /usr ro,logging
filesys rootdisk.s4 4096 /var logging,nosuid
filesys rootdisk.s7 10 unnamed
filesys rootdisk.s5 free /opt logging,nosuid
```

For this configuration, an initial install is performed and not an upgrade, and the client is a standalone server. The Core Solaris software is installed along with administrative packages, NTP, system accounting, and the gzip binary and compression libraries. The root disk is specified as c0t0d0s0 and the partitioning option explicit states that the disk layout is being specified with **filesys**. The sizes are specified for / (root), /usr, /var, and unnamed – which is used for DiskSuite. The swap file system is set to auto. The actual size is determined by the software packages being installed. The final entry tells the jumpstart client to use the rest of the disk for /opt. The entry **free** must be the last entry when specifying file systems. The options for each file system entry are put in /etc/vfstab and take effect after the client has had all the software installed and the system has rebooted.[38]

**Begin Scripts**

For this server, there are not any begin scripts. The most common begin scripts are used to create a backup of the system before it gets a new operating system installed.[39] An example of this would be:

```
#!/bin/sh
ufsdump 0f /dev/rmt/0 /dev/dsk/${SI_ROOTDISK}
```

This script was taken from Solaris Blueprint book.

**Finish Scripts**

These scripts are executed after the Operating System has been installed. The scripts are useful for installing patches, additional software, etc. The finish scripts for this system perform the same tasks that were done when building the jumpstart server. This helps to make sure that every client that has been jumpstarted is configured the same. For this jumpstart configuration, there is only one finish script, however it executes other scripts. This script creates the /usr/local directory, executes other scripts that perform tasks such as installing patches, and then runs the fix-modes program with the base directory set to /a. This is because while the operating system is being installed, all the mount points of the disk have a base directory of /a. So root is mounted as /a, var is mounted as /var, etc. All of theses tasks could be accomplished in one script, however a modular approach was taken and each script executed by the finish script has specific tasks.[40]

/opt/jumpstart/finish/finish.sh:

```
#!/bin/sh
# /opt/jumpstart/finish/finish.sh
# Finish Script that calls other scripts to finish the install
#
#
echo ""
echo "Starting finish scripts..."
echo ""
script_dir="${SI_CONFIG_DIR}/scripts"

mkdir /a/usr/local
chmod 755 /a/usr/local
script_list="installpatches.sh pkginstall.sh filecp.sh touchup.sh"

for script in $script_list
do
        if [ -f "$script_dir/$script" ]
        then
                echo "Starting finish script: $script"
                echo ""
                . $script_dir/$script
                echo "Finished: $script"
                echo ""
        else
                echo "ERROR: file not found: $script"
                echo ""
        fi
done

echo "Running fix-modes"
cd $script_dir/fix-modes
./fix-modes -R /a

echo "Jumpstart has finished. Resetting the system in 15 seconds..."
sleep 15
```

The installpatches.sh script checks the version of the operating system and then checks for the recommended patch cluster for the appropriate OS and starts the install. This script was taken from JumpStart Technology.[41]

```
#!/bin/sh
#
#ident "@(#)install-recommended-patches.fin  1.6  00/10/19 SMI"
#
# This script is responsible for installing a Sun Recommended
# and Security Patch Cluster from ${BASEDIR}/${PATCH_DIR}.

errorCondition=0
mountedProc=0

BASEDIR="/a"
PATCH_SERV_DIR=""
PATCH_DIR="/mnt"
MNTTAB="${BASEDIR}/etc/mnttab"
OE_VER="`uname -r`"

mount -F nfs -o ro 10.0.1.8:/jumpstart/Patches ${BASEDIR}/${PATCH_DIR}

case ${OE_VER} in

  5.8)
    PATCH_SERV_DIR=8_Recommended
    ;;

  5.7)
    PATCH_SERV_DIR=7_Recommended
    ;;

  5.6)
    PATCH_SERV_DIR=2.6_Recommended
    ;;

  5.5.1)
    PATCH_SERV_DIR=2.5.1_Recommended
    ;;

  *)
    errorCondition=1
    ;;

esac


if [ ${errorCondition} = 0 ]; then
  if [ ! -d ${BASEDIR}/${PATCH_DIR} ]; then
    echo "The directory, ${PATCH_DIR}, does not exist."
  else

    # Some patches require a loopback filesystem be used when
    # installing using chroot.

    if [ -d /proc ]; then
      if [ "`df -n /proc | awk '{ print $3 }'`" = "proc" ]; then
        if [ -d ${BASEDIR}/proc ]; then
          if [ "`df -n ${BASEDIR}/proc | \
            awk '{ print $3 }'`" != "proc" ]; then
            mount -F lofs /proc ${BASEDIR}/proc
            mountedProc=1
          fi
        fi
      fi
    fi

    if [ ! -s ${MNTTAB} ]; then
      if [ -s /etc/mnttab ]; then

        # First create ${MNTTAB} so patches can read it:

        echo "Copying /etc/mnttab from miniroot to ${MNTTAB}"
        echo ""

        rm -f ${MNTTAB}

        if [ "${OE_VER}" = "5.5.1" ]; then

          # This is necessary for "install_cluster" to get the mount
          # point for /var/sadm/patch from the "real" root filesystem.

          cat /etc/mnttab | sed 's/\/a/\//g' > ${MNTTAB}

          # This is necessary for "df" to execute which is needed by
          # "install_cluster" to determine if enough free disk
          # space exists on the target system.

          touch          ${BASEDIR}/etc/.mnttab.lock
          chown root:root ${BASEDIR}/etc/.mnttab.lock
          chmod 644       ${BASEDIR}/etc/.mnttab.lock
        else
```

```
        cp /etc/mnttab ${MNTTAB}
      fi
    else
      echo "Could not find a valid /etc/mnttab"
      errorCondition=1
    fi
  fi

  if [ ${errorCondition} = 0 ]; then

    SHOWCOMMAND=""

    if [ -x ${BASEDIR}/usr/sbin/patchadd ]; then
      SHOWCOMMAND="/usr/sbin/patchadd"
    elif [ -x ${BASEDIR}/usr/bin/showrev ]; then
      SHOWCOMMAND="/usr/bin/showrev"
    fi

    cd ${BASEDIR}/${PATCH_DIR}

    if [ -d ${PATCH_SERV_DIR} ]; then
      echo "Installing the ${PATCH_SERV_DIR} patch cluster."
      echo ""

      if [ "${SHOWCOMMAND}" = "/usr/sbin/patchadd" ]; then
        chroot ${BASEDIR} /usr/sbin/patchadd -d -u \
          -M ${PATCH_DIR}/${PATCH_SERV_DIR} patch_order
      elif [ -x ${PATCH_DIR}/${PATCH_SERV_DIR}/install_cluster ]; then
        chroot ${BASEDIR} \
          ${PATCH_DIR}/${PATCH_SERV_DIR}/install_cluster -q \
          ${PATCH_DIR}/${PATCH_SERV_DIR}
      else
        echo "Cannot find /usr/sbin/patchadd or install_cluster"
      fi
    else
      echo "Could not find the ${PATCH_SERV_DIR} patch cluster"
    fi
  fi

  umount ${BASEDIR}/${PATCH_DIR}
  if [ ${mountedProc} = 1 ]; then
    umount ${BASEDIR}/proc
  fi
fi
```

```
fi
```

The pkginstall.sh script is used to install additional packages. This script does not automatically install any package that is in the /opt/jumpstart/package directory. Instead, only specific packages are installed. If additional packages are desired, the administrator will have to modify the script. For the scripts to not prompt the adminstrator for any input, an administration file needs to be created.[42]

/opt/jumpstart/packages/noask:

```
mail=
instance=overwrite
partial=nocheck
runlevel=nocheck
idepend=nocheck
rdepend=nocheck
space=nocheck
setuid=nocheck
conflict=nocheck
action=nocheck
basedir=default
```

/opt/jumpstart/scripts/pkginstall.sh:

```
#!/bin/sh
# This script installs specific packages. If you add any packages,
# you MUST update this script

BASEDIR=/a
ADMINFILE=${SI_CONFIG_DIR}/packages/noask
PKGDIR=${SI_CONFIG_DIR}/packages

echo "Installing OpenSSH..."
/usr/sbin/pkgadd -a $ADMINFILE -d $PKGDIR/OpenSSH.pkg \                    -
R $BASEDIR OpenSSH
echo "Installing Tara..."
/usr/sbin/pkgadd -a $ADMINFILE -d $PKGDIR/ARCtara.pkg \                    -
R $BASEDIR ARCtara
echo "Installing GNUpg..."
/usr/sbin/pkgadd -a $ADMINFILE -d $PKGDIR/GNUpg.pkg \                      -
R $BASEDIR GNUpg
echo "Installing ipfilter..."
/usr/sbin/pkgadd -a $ADMINFILE -d $PKGDIR/ipf.pkg \                        -
R $BASEDIR ipfx ipf
echo "Installing logcheck..."
/usr/sbin/pkgadd -a $ADMINFILE -d $PKGDIR/PSlogchk.pkg \                   -
R $BASEDIR logchk
echo "Installing lsof..."
/usr/sbin/pkgadd -a $ADMINFILE -d $PKGDIR/PUlsof.pkg \                     -
R $BASEDIR PUlsof
echo "Installing snort..."
/usr/sbin/pkgadd -a $ADMINFILE -d $PKGDIR/snort.pkg \                      -
R $BASEDIR Snort
echo "Installing tripwire..."
/usr/sbin/pkgadd -a $ADMINFILE -d $PKGDIR/tripwire.pkg \                   -
R $BASEDIR tripwire
echo "Installing sudo..."
/usr/sbin/pkgadd -a $ADMINFILE -d $PKGDIR/sudo.pkg -R $BASEDIR sudo
```

filecp.sh copies files from the /opt/jumpstart/files directory and puts them in the place specified within the script. As with the package installation script, the script does not automatically copy every file over, each file must have an entry in the script for the file to be copied over. The files to be are copied are the files that were edited or created when the jumpstart server was built. A copy of each of those files has been put in /opt/jumpstart/files. The ssh host keys are created on the Jumpstart server by the addclient script, these keys are also copied to the client. The last file to be copied over performs tasks that can only be done after the system has rebooted. For this configuration, that script removes unwanted users and modifies the default shell for others. A table of files copied to the jumpstart client is in Appendix B.

/opt/jumpstart/scripts/filecp.sh:

```
#!/bin/sh
#
# File copy script for finishing up the install
#

echo "Copying configuration files..."

# netconfig
cp -p  ${SI_CONFIG_DIR}/files/netconfig /a/etc/init.d/netconfig
cd /a/etc/rc2.d
ln -s ../init.d/netconfig S69netconfig

# defaultrouter disable ip forwarding
cp -p  ${SI_CONFIG_DIR}/files/defaultrouter /a/etc/defaultrouter
touch /a/etc/notrouter

# New inetsvc
cp -p  ${SI_CONFIG_DIR}/files/newinetsvc /a/etc/init.d/newinetsvc
cd /a/etc/rc2.d
rm S72inetsvc
ln -s ../init.d/newinetsvc S72inetsvc

# New syslog
cp -p  ${SI_CONFIG_DIR}/files/newsyslog /a/etc/init.d/newsyslog
cd /a/etc/rc2.d
rm S74syslog
ln -s ../init.d/newsyslog S74syslog

# /etc/inittab
cp -p  ${SI_CONFIG_DIR}/files/inittab /a/etc/inittab

# sendmail
cp -p  ${SI_CONFIG_DIR}/files/sendmail /a/etc/init.d/sendmail
cp -p  ${SI_CONFIG_DIR}/files/sendmail.cf /a/etc/mail/sendmail.cf
echo "# Flush mail queue" >> /a/var/spool/cron/crontabs/root
echo "0 * * * * /usr/lib/sendmail -q 2>&1" >> /a/var/spool/cron/crontabs/root

# ntp
```

```
cp -p ${SI_CONFIG_DIR}/files/ntp.conf /a/etc/inet/ntp.conf
touch /a/etc/inet/ntp.driftfile

# resolv.conf and nsswitch
cp -p  ${SI_CONFIG_DIR}/files/resolv.conf /a/etc/resolv.conf
cp -p  ${SI_CONFIG_DIR}/files/nsswitch.conf /a/etc/nsswitch.conf

# sshd_config
cp -p  ${SI_CONFIG_DIR}/files/sshd_config /a/usr/local/etc/sshd_config

# tcp wrappers files
cp -p  ${SI_CONFIG_DIR}/files/hosts.allow /a/etc/hosts.allow
cp -p  ${SI_CONFIG_DIR}/files/hosts.deny /a/etc/hosts.deny

# ftpusers
cp -p  ${SI_CONFIG_DIR}/files/ftpusers /a/etc/ftpusers

# pam.conf
cp -p  ${SI_CONFIG_DIR}/files/pam.conf /a/etc/pam.conf

# cron and at
cp -p  ${SI_CONFIG_DIR}/files/at.allow /a/etc/cron.d/at.allow
cp -p  ${SI_CONFIG_DIR}/files/cron.allow /a/etc/cron.d/cron.allow
rm -rf /a/etc/cron.d/*.deny

# syslog
cp -p  ${SI_CONFIG_DIR}/files/syslog.conf /a/etc/syslog.conf

# Log rotation
cp -p  ${SI_CONFIG_DIR}/files/rotatelog /a/usr/local/bin
echo "# Logrotation scripts" >> /a/var/spool/cron/crontabs/root
echo "3 0 * * 0 /usr/local/bin/rotatelog /var/log/authlog y 2>&1" >> \
        /a/var/spool/cron/crontabs/root
echo "5 0 * * 0 /usr/local/bin/rotatelog /var/log/loginlog y 2>&1" >> \
        /a/var/spool/cron/crontabs/root
echo "7 0 * * 0 /usr/local/bin/rotatelog /var/log/syslog y 2>&1" >> \
        /a/var/spool/cron/crontabs/root
echo "9 0 * * 0 /usr/local/bin/rotatelog /var/log/snort/alert y 2>&1" >> \
        /a/var/spool/cron/crontabs/root
echo "11 0 * * 0 /usr/local/bin/rotatelog /var/log/snort/portscan.log 2>&1" >> \
        /a/var/spool/cron/crontabs/root

# System accounting
cp -p  ${SI_CONFIG_DIR}/files/perf /a/etc/init.d/perf
cp -p  ${SI_CONFIG_DIR}/files/sa1 /a/usr/lib/sa/sa1
cp -p  ${SI_CONFIG_DIR}/files/sa2 /a/usr/lib/sa/sa2
cp -p  ${SI_CONFIG_DIR}/files/sys.crontab /a/var/spool/cron/crontabs/sys

# warning files
cp -p  ${SI_CONFIG_DIR}/files/motd /a/etc/motd
cp -p  ${SI_CONFIG_DIR}/files/issue /a/etc/issue

# /etc/default
cp -p  ${SI_CONFIG_DIR}/files/inetinit /a/etc/default/inetinit
cp -p  ${SI_CONFIG_DIR}/files/cron /a/etc/default/cron
cp -p  ${SI_CONFIG_DIR}/files/su /a/etc/default/su
cp -p  ${SI_CONFIG_DIR}/files/passwd /a/etc/default/passwd
cp -p  ${SI_CONFIG_DIR}/files/init /a/etc/default/init
cp -p  ${SI_CONFIG_DIR}/files/login /a/etc/default/login

# Host keys
cp -p ${SI_CONFIG_DIR}/files/`uname -n`.ssh_host_key /a/usr/local/etc/ssh_host_key
cp -p ${SI_CONFIG_DIR}/files/`uname -n`.ssh_host_key \
        /a/usr/local/etc/ssh_host_key.pub
cp -p ${SI_CONFIG_DIR}/files/`uname -n`.ssh_host_dsa_key \
        /a/usr/local/etc/ssh_host_dsa_key
cp -p ${SI_CONFIG_DIR}/files/`uname -n`.ssh_host_dsa_key \
        /a/usr/local/etc/ssh_host_dsa_key.pub
cp -p ${SI_CONFIG_DIR}/files/`uname -n`.ssh_host_rsa_key \
        /a/usr/local/etc/ssh_host_rsa_key
cp -p ${SI_CONFIG_DIR}/files/`uname -n`.ssh_host_rsa_key \
        /a/usr/local/etc/ssh_host_rsa_key.pub

# ipfilter
cp -p ${SI_CONFIG_DIR}/files/ipf.conf /a/etc/opt/ipf/ipf.conf

# md5 hash binary
cp -p ${SI_CONFIG_DIR}/files/md5 /a/usr/local/bin

# Final touch-up script that runs after reboot
cp -p  ${SI_CONFIG_DIR}/files/S99finalcleanup /a/etc/rc3.d/S99finalcleanup
```

The last script to be executed performs miscellaneous touchups. Some of the functions the script performs are renaming unnecessary startup files in /etc/rc2.d so certain services don't start at system boot, removing inetd.conf since it is not used, creating log files, etc.

/opt/jumpstart/scripts/touchup.sh

```
#!/bin/sh
#
# Misc touch ups here
#
# Remove these from system boot

echo "Performing final touch ups..."

echo "Removing startup files..."
for file in S30sysid.net S71sysid.sys S72autoinstall S71rpc S76nscd S73nfs.client
S74autofs S73cachefs.daemon S71ldap.client S80PRESERVE
do
        mv /a/etc/rc2.d/$file /a/etc/rc2.d/.off.$file
done

mv /a/etc/rc3.d/S15nfs.server /a/etc/rc3.d/.off.S15nfs.server

# Removing unnecessary files

echo "Removing unnecessary files..."

# inetd
rm /a/etc/inet/inetd.conf /a/etc/inetd.conf
# NFS
rm /a/etc/auto_home /a/etc/auto_master /a/etc/dfs/dfstab
# crontab
rm /a/var/spool/cron/crontabs/adm /a/var/spool/cron/crontabs/lp

# Creating empty r-files
for file in /a/.rhosts /a/.shosts /a/.netrc /a/etc/hosts.equiv
do
        cp /a/dev/null $file
        chown root:root $file
        chmod 000 $file
done

# Creating log files
touch /a/var/log/authlog
chown root /a/var/log/authlog
chmod 600 /a/var/log/authlog

touch /a/var/adm/loginlog
```

```
chown root:sys /a/var/adm/loginlog
chmod 600 /a/var/adm/loginlog

# eeprom settings - not password though...
eeprom oem-banner="Authorized users only. \
 All access may be logged and reported. "
eeprom oem-banner\?=true
```

**Rules**

The rules file is used to define the system hostname, the begin script to run, the profile to use, and the finish script to run. An example of a rules file is:

/opt/jumpstart/rules

```
# Rules file created: Thu Dec 20 20:56:13 GMT 2001
# Created by addclient script
hostname smithers - profiles/desktop.profile finish/finish.sh
```

The dash states that there is no begin script to be used. The jumpstart process does not actually read the rules file. Instead it uses the **rules.ok** file.[43] This file is generated by the **check** script. The check script can be found on the Solaris 8 media. Place a copy of this script in the /opt/jumpstart directory. If any modifications are made to the rules file, the check script must be executed again. Note that this task will be done when the **addclient** script is ran.

**addclient**

This script is a derived from /jumpstart/OS/Solaris_8/Tools/add_install_client. The essentials of this script are that given the hostname, IP address, MAC address, and system type, the appropriate configuration changes will be made to allow a jumpstart from the new client. The script follows these basic steps:

1. Check options and make sure all required fields are populated. Note that these data are not checked for errors.
2. Check to see if there are current entries for a system of the same IP address. If there are, check to see if the force option is set. If the option is set, remove the conflicting **at** job and continue executing the script, otherwise exit with an error.
3. Build the appropriate rules and rules.ok file.
4. Add entries to /etc/hosts, /etc/ethers, and /etc/bootparams. The ethers and bootparams files are recreated each time this script is run. For the hosts file, the script copies the original hosts file and then appends the hosts information for the jumpstart client.
5. Calculates the hexadecimal IP address. This is used by tftpboot.
6. Unmounts /tftpboot since it is read only and removes the directory. The script

then mounts it under /opt/jumpstart/mnt and creates the files required to boot the jumpstart client. The file system is then unmounted, the /tftpboot directory is recreated, and the file system is mounted to /tftpboot as read only.

7. Creates the OpenSSH host keys and places them in the /opt/jumpstart/files directory. This is done here because when creating the keys, OpenSSH depends on system entropy and new system will have very little system entropy.
8. IP filter rules are put in place to allow the system access to the jumpstart server.
9. An **at** job is created to remove the IP filter rules and it is scheduled to happen in four hours. The /tftpboot file system is also unmounted, mounted under /opt/jumpstart/mnt, cleaned of all files, and remounted as read-only under /tftpboot.

Before executing this script, be sure to create a copy of /etc/hosts called /etc/hosts.orig.

/opt/jumpstart/addclient:

```
#!/bin/ksh
#
# Initializing the FORCE variable
FORCE=0
# Get arguments here

while getopts "n:i:e:t:f" opt;
do
        case $opt in
                n ) CLIENT=$OPTARG;;
                i ) IP_ADDR=$OPTARG;;
                e ) MAC_ADDR=$OPTARG;;
                t ) MACH_TYPE=$OPTARG;;
                f ) FORCE=1;;
                \? ) echo 'usage addclient -n hostname -i IP address -e MAC Address –t
[server|xserver|desktop] -f';;
        esac
done

# Checking the arguments here. If there is an error, echo the usage.
error=0

if [ -z $CLIENT ];
then
        error=1
elif [ -z $IP_ADDR ];
```

```
then
        error=1
elif [ -z $MAC_ADDR ];
then
      error=1
elif [ -z $MACH_TYPE ];
then
        error=1
elif [ $MACH_TYPE != "server" ] && [ $MACH_TYPE != "xserver" ] && \
        [$MACH_TYPE != "desktop" ];
then
        error=1
fi

if [ $error -eq 1 ];
then
        echo 'usage addclient -n hostname -i IP address -e MAC Address –t
[server|xserver|desktop] -f'
        exit 255
fi

# Look for at jobs per IP address, if you find one, remove it as long as the force
# option is set

atdir=/var/spool/cron/atjobs

for job in `at -l | awk '{ print $4 }'`
do
        if /usr/xpg4/bin/grep -F $IP_ADDR $atdir/$job > /dev/null;
        then
                if [ $FORCE = "1" ];
                then
                        echo "Removing $job"
                        at -r $job
                else
                        echo "Warning: $IP_ADDR is already in the jumpstart cleanup."
                        echo "        Use "-f" to force the script to re-add the client"
                        exit 255
                fi
        fi
done

# Now we start to build the rules file and check it.
```

```
JSPATH=/opt/jumpstart
BOOTFILE=${JSPATH}/OS/Solaris_8/Tools/Boot/usr/platform/sun4u/lib/fs/nfs/inetboot

DATE=`date`
echo "# Rules file created: $DATE" > $JSPATH/rules
echo "# Created by addclient script" >> $JSPATH/rules

if [ $MACH_TYPE == "server" ]; then
        echo "hostname $CLIENT - profiles/server_large.profile finish/finish.sh" >>
$JSPATH/rules
        echo "hostname $CLIENT - profiles/server_small.profile finish/finish.sh" >>
$JSPATH/rules
elif [ $MACH_TYPE == "xserver" ]; then
        echo "hostname $CLIENT - profiles/xserver_large.profile finish/finish.sh" >>
$JSPATH/rules
        echo "hostname $CLIENT - profiles/xserver_small.profile finish/finish.sh" >>
$JSPATH/rules
else
        echo "hostname $CLIENT - profiles/desktop.profile finish/finish.sh" >>
$JSPATH/rules
fi

if [ -x $JSPATH/check ]; then
        cd $JSPATH
        ./check
        if [ $? -ne 0 ];
        then
                echo "Rule check failed, exiting..."
                exit 255
        fi
fi

# Copying the original hosts file and
# adding the new system to the hosts table
if [ ! -f /etc/hosts.orig ];
        then
        echo "/etc/hosts.orig doesn't exist. Please create one with at least"
        echo "this systems IP address and hostname"
        exit 255
fi

cp /etc/hosts.orig /etc/hosts

# Adding the host info to /etc/ethers
```

```
echo "$IP_ADDR       $CLIENT        ${CLIENT}.bankone.net" >> /etc/hosts
echo "$MAC_ADDR $CLIENT" > /etc/ethers

# Creating /etc/bootparams
echo "$CLIENT  root=loki-js:${JSPATH}/OS/Solaris_8/Tools/Boot install=loki-
js:${JSPATH}/OS boottype=:in sysid_config=loki-js:${JSPATH}/sysidcfg
install_config=loki-js:${JSPATH} rootopts=:rsize=32768" > /etc/bootparams

# Translating decimal IP to Hex
HEXIP=`echo $IP_ADDR |\
awk -F. '{
        IP_HEX = sprintf("%0.2x%0.2x%0.2x%0.2x", $1,$2,$3,$4)
        print IP_HEX
        }' | tr '[a-z]' '[A-Z]'`

# Unmounting /tftpboot, error if busy
umount /tftpboot
if [ $? -ne 0 ];
then
            echo "/tftpboot is busy, couldn't unmount"
            exit 255
fi

# Removing directory in case some one decides to write to it.
rm -rf /tftpboot

# Checking to see if jumpstart mount point is available
if [ ! -d ${JSPATH}/mnt ];
then
            echo "Error: ${JSPATH}/mnt is a file"
            echo "Please make it an emtpy directory"
            exit 255
fi

# Mount the tftpboot disk
mount -o rw /dev/md/dsk/d6 ${JSPATH}/mnt

# Remove old files
for file in `ls ${JSPATH}/mnt | grep -v inetboot.SUN4U.Solaris_8-1 | grep -v lost`
do
            rm ${JSPATH}/mnt/$file > /dev/null
done

# Compare the current bootfile with the proper one
# Copy over the current if there is a difference
cmp -s ${JSPATH}/mnt/inetboot.SUN4U.Solaris_8-1 $BOOTFILE

if [ $? -ne 0 ];
then
        cp $BOOTFILE ${JSPATH}/mnt/inetboot.SUN4U.Solaris_8-1
else
        touch ${JSPATH}/mnt/inetboot.SUN4U.Solaris_8-1
fi

chmod 755 ${JSPATH}/mnt/inetboot.SUN4U.Solaris_8-1

# Create the links for HEXIP boot file and HEXIP.SUNW4U
if [ -f ${JSPATH}/mnt/$HEXIP ];
then
        touch ${JSPATH}/mnt/$HEXIP
else
        cd ${JSPATH}/mnt
        ln -s  inetboot.SUN4U.Solaris_8-1 $HEXIP
fi

if [ -f ${JSPATH}/mnt/$HEXIP.SUN4U ];
then
        touch ${JSPATH}/mnt/$HEXIP.SUN4U
else
        cd ${JSPATH}/mnt
        ln -s  inetboot.SUN4U.Solaris_8-1 $HEXIP.SUN4U
fi

cd
# Unmount the tftpboot from the jumpstart mount, error if busy

umount ${JSPATH}/mnt
if [ $? -ne 0 ];
then
        echo "Error: Couldn't unmount ${JSPATH}/mnt"
        exit 255
fi

mkdir /tftpboot
mount /tftpboot
if [ $? -ne 0 ];
then
        echo "Error: Couldn't mount /tftpboot"
        exit 255
fi

# Create the host keys
bindir=/usr/local/bin
sysconfdir=${JSPATH}/files

OLDHOST=`uname -n`
hostname $CLIENT
$bindir/ssh-keygen -q -t rsa1 -f $sysconfdir/$CLIENT.ssh_host_key -N ""
$bindir/ssh-keygen -q -t dsa -f $sysconfdir/$CLIENT.ssh_host_dsa_key -N ""
$bindir/ssh-keygen -q -t rsa -f $sysconfdir/$CLIENT.ssh_host_rsa_key -N ""
hostname $OLDHOST

# Add ipf rules for the system to boot

echo "pass in quick on hme1 proto icmp from $IP_ADDR/32 to loki-js/32" | ipf -f -
echo "pass in quick on hme1 proto tcp/udp from $IP_ADDR/32 to loki-js/32"\
        | ipf -f –

# Schedule the at job to add the firewall rules and remove the links from tftpboot  This
will die if /tftpboot is busy

at now + 4 hour << EOF

echo "pass in quick on hme1 proto icmp from $IP_ADDR/32 to loki-js/32" | ipf -rf -
echo "pass in quick on hme1 proto tcp/udp from $IP_ADDR/32 to loki-js/32 " \
        | ipf -rf -

umount /tftpboot
if [ $? -ne 0 ];
then
        echo "Warning: Couldn't unmount /tftpboot, it is busy. "
        echo "Exiting..."
        exit 255
fi

mount -o rw /dev/md/dsk/d6 ${JSPATH}/mnt
rm $HEXIP $HEXIP.SUN4U
umount ${JSPATH}/mnt
mount /tftpboot

rm -rf ${sysconfdir}/${CLIENT}.ssh_host_key*
rm -rf ${sysconfdir}/${CLIENT}.ssh_host_dsa_key*
rm -rf ${sysconfdir}/${CLIENT}.ssh_host_rsa_key*

EOF
```

Note that two systems can be jumpstarted at the same time. However, two systems cannot in the process of booting at the same time. This is a limitation of this script and not jumpstart itself. The issue is that /tftpboot is mounted read-only and to create the necessary boot files for client2, /tftpboot needs to be un-mounted and re-mounted as read-write. If client1 is in the process of booting, /tftpboot will either be busy, or client1 will fail to boot because /tftpboot is un-mounted causing client1to no longer have access to the necessary boot file.

**Final Touchup**

Here are the last few steps that are required before the system can be put in production.

**Initialize Tripwire**

Even if this was done earlier, initialize the database again. The reason for this is all the additional files, scripts, and directories that have been added. If this is not done, a huge email will be sent the next time Tripwire runs an integrity check.

```
# cd /var/tripwire
# /usr/local/bin/tw/tripwire -initialize -c /etc/opt/tw/tw.config
# …output deleted…
# mv databases/tw.db_$myhost /var/tripwire
```

**Make a backup**

Using the attached tape drive, create a backup of all file systems. Store this in a secure area and leave it. If any significant changes are made, then make another backup using a new tape. Be sure to properly dispose of any old backup media.

## Performing a Jumpstart

Jumpstarting a client is a very simple task. The first step is to determine the client's hostname, IP address, and the profile to use. Then physically assemble the client and place it in the secured area. Connect it to the 192.168.200.0 network, attach a terminal connection or keyboard and monitor, and power on the system. (If an operating system is currently on the client, break out of the system boot.) At the EEPROM prompt enter:

```
{2} ok banner
```

If a customized banner is displayed, re-enable the default banner with:

```
{2} ok setenv oem-banner? false
```

Then re-issue the **banner** command. This prints out the MAC address for the client which is need to boot the client over the network. Now on the Jumpstart server as root (or with sudo) enter:

```
# ./addclient -n client1 -i 192.168.200.x -e MAC address from above -t server
Validating rules...
Validating profile profiles/server_large.profile...
Validating profile profiles/server_small.profile...
The custom JumpStart configuration is ok.
commands will be executed using /usr/bin/ksh
job 1009422388.a at Sun Dec 23 03:06:28 2001
```

Now on the client, issue the following command to start the install:

```
{2} ok boot net – install
Resetting ...
…output omitted…
Boot device: /pci@1f,4000/network@1,1  File and args: - install
Timeout waiting for ARP/RARP packet
2aa00
SunOS Release 5.8 Version Generic_108528-09 64-bit
Copyright 1983-2001 Sun Microsystems, Inc.  All rights reserved.
…output omitted…
Using sysid configuration file 192.168.200.1:/opt/jumpstart/sysidcfg/sysidcfg
Checking rules.ok file...
Using profile: profiles/server_large.profile
Using finish script: finish/finish.sh
Executing JumpStart preinstall phase...
Searching for SolStart directory...
Checking rules.ok file...
Using begin script: install_begin
Using finish script: patch_finish
Executing SolStart preinstall phase...
Executing begin script "install_begin"...
Begin script install_begin execution completed.
…output omitted…
```

After the installation has completed, the client will reboot.

**Post Jumpstart Tasks**

There are still some tasks that the administrator needs to complete after a client jumpstart has completed.

1. Edit the IP filter rules. Make sure to add the appropriate entries in /etc/opt/ipf/ipf.conf. The jumpstart does not put a valid configuration file in place. All traffic will be blocked, but no inbound traffic, including ssh, will be accepted.
2. Set the password on the EEPROM.
3. If the system has multiple interfaces, add all the to the HOME_NET entries to /etc/opt/snort/snort.conf. For example for the jumpstart server, the entry is:
   var HOME_NET [loki/24,loki-js/24]
4. Test and test some more. Be sure to check that the system functions as planned. Some of the tests are checking to see if a file can be written to /usr, run lsof –i to show what ports the system is listening on, etc.

## Ongoing Maintenance

First things first, sign up to the announcement mailing lists for each of the software products that are installed. Note that some may not have just an announcement mailing list. There are other security mail lists that an administrator should subscribe to such as Cert Advisory Mailing List at http://www.cert.org, BUGTRAQ at http://www.securityfocus.com, and the SunSolve Patch Club at http://www.sunsolve.com. Also watch the newsgroup comp.security.announce.

### Backups

The only data on this system that are worth backing up are the scripts used to jumpstart systems. Since the operating system images can be retrieved via the installation media, consider backing up /opt/jumpstart minus the directory OS. More than likely the scripts won't change much, so a manual backup would be adequate. This could be done by tape or over the network via scp.

### Patches

There are only a few times when any of the software on this system should be patched. The first reason is when a security hole is found. When a hole is detected, immediately download and install the necessary patch. If a fix is not available, then consider shutting down the service altogether until a fix is released.

Another reason is based upon reliability. If the system or software is crashing on a regular basis, consider upgrading, downgrading, or patching the software. While building this server, the latest version of IP Filter (3.4.22) would crash without warning and without logging any information. Upon browsing the IP Filter mailing lists, it was determined that Solaris 8 and this version do not play well and the last version that runs reliably on Solaris 8 is version 3.4.20.

Consider checking the latest recommended patches on a regular bases and putting them in the /opt/jumpstart/patches directory. That way any system built will have the latest patches installed at that time. After a patch has been installed, be sure to update the tripwire database. Otherwise a tripwire integrity violation will be noted and an email will be sent. Depending on the size of the patch, this email could be huge.

### Scanners

Scanning the host is an effective way of protecting the system against attacks. By using these tools below, the administrator can make it tougher for an attacker to successfully break in to the system. By making it difficult for an attacker, the end result may cause the attacker to give up and move on to an easier target.

### Tara

Tara is used to scan the host for a number of issues, such as world writeable directories, questionable user accounts, etc. As in the other packages that have been created, a cron job will be put in place when the package is installed.

```
0 5 * * * /usr/local/tiger/tiger > /dev/null 2>&1
```

When the script runs, the output is stored in /var/tiger/logs. An example of the logs is below.

```
Security scripts *** undetermined ***
Mon Dec 17 00:05:43 GMT 2001
00:05> Beginning security report for loki (sun4u SunOS 5.8).

# Performing check of passwd files...

# Performing check of group files...

# Performing check of user accounts...
# Checking accounts from /etc/passwd.
--WARN-- [acc005w] Login ID sys is disabled, but has a 'cron' file or cron
        entries.
--WARN-- [acc006w] Login ID adm's home directory (/var/adm) has group `sys'
        write access.

# Performing check of /etc/hosts.equiv and .rhosts files...

# Checking accounts from /etc/passwd...

# Performing check of .netrc files...

# Checking accounts from /etc/passwd...

# Performing check of anonymous FTP...

# Performing checks of mail aliases...
# Checking aliases from /etc/mail/aliases.

# Performing check of 'services' and 'inetd'...
# Checking services from /etc/services.
--FAIL-- [inet003f] The port for service pop-2 is assigned to service pop2.
--ERROR-- [init005e] Don't have required file INETDFILE.

# Performing NFS exports check...
--WARN-- [nfs002w] Anonymous ID == 0 for R/O filesystem /opt/jumpstart
--WARN-- [nfs007w] Directory /opt/jumpstart exported R/O to everyone.

# Performing system specific checks...
# Performing checks for SunOS/5...
```

```
--WARN-- [misc008w] NFS port checking disabled in kernel.
--ERROR-- [misc005w] Can't find check_sendmail'...
```

Note that there are some false positives such as the message about the user sys. The user sys has cron entries for system accounting and the account is locked because no one needs to log as sys. So that message and similar messages can be ignored. The warning about exporting /opt/jumpstart anonymously cannot be avoided, it is required by Jumpstart.

**NMAP**

This utility should be run from another system on the same network. Otherwise a firewall may be blocking access or the scan may set off an alarm on a Intrusion Detection System. It is used to verify that only the appropriate ports are listening on the system. Since this system is using services that require UDP and TCP, scans of both protocols will be performed. Pings will not be performed because the scan will not complete if the ping fails, and the Jumpstart server has ICMP blocked which would cause the ping to fail and the scan will exit.

```
# nmap –sT –sU –P0 loki
```

Consider running this on a weekly basis via a script and having the output parsed, if any ports other than sshd are listening, then trigger an alarm.

**System Monitoring**

**Tripwire**

After the Jumpstart server had been built and configured the Tripwire database was initialized. This created a system baseline of the files being watched. If these files are changed outside of the checks performed on it, the system administrators will be informed. If tripwire was installed with the package created earlier, then a cron job should already exists. If changes have been made to the configuration file, the database needs to be re-initialized. This initialization is done by:

```
# cd /var/tripwire
# /usr/local/bin/tw/tripwire -initialize -c /etc/opt/tw/tw.config
# mv databases/tw.db_loki /var/tripwire
# rmdir databases
```

Verify the cron entry:

```
0 10 * * * /usr/local/bin/tw/tripwire -c /etc/opt/tw/tw.config 2>&1 | /usr/bin/mailx –s \
        'Tripwire output from `uname -n`' root@destination.com
```

Now when tripwire runs an integrity check, if there are any violations, root will be notified via email. An example would an attacker placing a Trojan horse to be installed in all the jumpstarted systems.

**Logcheck**

This utility searches the logs for any inconsistencies and reports them back to root via email. The logcheck package that was created and install in the Additional Software section also adds an entry into roots crontab that runs hourly.

```
0 * * * * /usr/local/bin/logcheck.sh 2>&1
```

The output below is an excerpt from one message to root.

```
Security Violations
=-=-=-=-=-=-=-=-=-=-=
Dec 23 00:03:21 loki su: [ID 366847 auth.notice] 'su root' succeeded for huffner on \
        /dev/pts/1

Unusual System Events
=-=-=-=-=-=-=-=-=-=-=-=-=
[**] [1:634:1] SCAN Amanda client version [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/20-06:13:21.841427 10.1.2.9:857 -> 10.1.1.1:10080
UDP TTL:253 TOS:0x0 ID:15543 IpLen:20 DgmLen:209 DF
Len: 189
[**] [1:498:2] ATTACK RESPONSES id check returned root [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
12/23-02:20:33.144456 192.168.200.1:2049 -> 192.168.200.55:1022
TCP TTL:64 TOS:0x0 ID:63519 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x346C3B4E  Ack: 0x22B1F6D  Win: 0x60F4  TcpLen: 20
```

There may be a lot of false positives, such as the second event listed under Unusual System events. The traffic appears to be from the client being jumpstarted. If that is the case, edit the appropriate configuration files in /usr/local/etc or for snort alerts edit /etc/opt/snort/snort.conf. (Any changes to the logcheck configuration will a require a reboot because /usr is mounted read only.) If in doubt, it is better to side on excess logs than not logging enough. However, note that too many false positives is like crying wolf, at some point administrators will begin to ignore the messages.

Again if an alarm goes of if and looks questionable, start the analysis. This could include port scans from systems, a large amount of failed login attempts or su attempts, or attempted connections from systems that don't have permission.

## Configuration Test

### System Verification

Apply the Solaris Practicum testing and verification process on the new Jumpstart server.[44] The following tests will be performed:

1. Able to ssh to system
2. Unable to ssh due to TCP Wrapper blocking.
3. Unable to login as root via ssh.
4. List of processes running and ports listening
5. Unable to write in /usr
6. Unable to run Set-UID from /var/tmp
7. Verify IP Filter rules
8. Checking the logs

### Can get here from there

Testing the ability to ssh to the new system.

```
$ ssh loki
The authenticity of host 'loki (10.1.1.1)' can't be established.
RSA key fingerprint is 52:69:55:4a:b9:53:d6:20:93:37:5f:b5:dc:fc:dc:9d.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts
(/export/home/huffner/.ssh/known_hosts).
huffner@loki's password:
Permission denied, please try again.
huffner@loki's password:
Last login: Sun Dec 23 20:18:17 2001 from 10.1.1.111
#############################################
…Motd omitted…
$
```

### Can't Get Here From There

Testing the TCP Wrappers denial of service from disallowed system.

```
huffner@badhost> ssh loki
ssh_exchange_identification: Connection closed by remote host
```

On the remote server the mail to root shows:

```
From: Super-User <root>
To: root@yourdomain.com
Subject: sshd@blah: connection attempt from 10.1.2.1
```

### Can't Get Here As Superuser

Testing ssh to host as root.

```
$ ssh -l root loki
root@loki's password:
Permission denied, please try again.
root@loki's password:
Permission denied, please try again.
root@loki's password:
Permission denied (publickey,password,keyboard-interactive).
```

**What's Running**

Listing of what is running and what ports are listening. Notice that are quite a few extra processes running and ports open that are required jumpstart.

```
$ ps -ef
     UID   PID  PPID  C    STIME TTY      TIME CMD
    root     0     0  0 20:33:08 ?        0:01 sched
    root     1     0  0 20:33:09 ?        0:00 /etc/init -
    root     2     0  0 20:33:09 ?        0:00 pageout
    root     3     0  0 20:33:09 ?        0:01 fsflush
    root   287     1  0 20:33:43 console  0:00 /usr/lib/saf/ttymon -g -h -p loki console \
           login: -T sun -d /dev/consle
    root   279     1  0 20:33:43 ?        0:00 /usr/lib/nfs/nfsd -a 16
    root   292   242  0 20:48:27 ?        0:01 /usr/local/sbin/sshd
    root    58     1  0 20:33:21 ?        0:00 /usr/lib/sysevent/syseventd
    root    60     1  0 20:33:21 ?        0:00 /usr/lib/sysevent/syseventconfd
    root   153     1  0 20:33:31 ?        0:00 /usr/sbin/inetd -s -t
    root   115     1  0 20:33:30 ?        0:00 ipmon -s
    root   277     1  0 20:33:42 ?        0:00 /usr/lib/nfs/mountd
    root   156     1  0 20:33:31 ?        0:00 /usr/local/sbin/rpcbind
    root   299   295  0 20:49:10 pts/1    0:00 ps -ef
    root   260     1  0 20:33:41 ?        0:00 /usr/lib/inet/xntpd
    root   181     1  0 20:33:32 ?        0:00 /usr/sbin/cron
    root   183     1  0 20:33:32 ?        0:00 /usr/sbin/syslogd -t
    root   247     1  0 20:33:40 ?        0:01 /usr/local/bin/snort -i hme1 -d -o -D –c \
           /etc/opt/snort/snort.conf
    root   201     1  0 20:33:33 ?        0:00 /usr/lib/utmpd
    root   242     1  0 20:33:38 ?        0:02 /usr/local/sbin/sshd
    root   206     1  0 20:33:35 ?        0:00 /usr/lib/sendmail -q15m
    root   245     1  0 20:33:40 ?        0:01 /usr/local/bin/snort -i hme0 -d -o -D –c \
           /etc/opt/snort/snort.conf
    root   282     1  0 20:33:43 ?        0:00 /usr/sbin/in.rarpd -a
    root   284     1  0 20:33:43 ?        0:00 /usr/sbin/rpc.bootparamd
# lsof –i
COMMAND   PID USER   FD   TYPE        DEVICE SIZE/OFF NODE NAME
inetd     153 root   11u  IPv6 0x300015e9970      0t0  UDP *:tftp (Idle)
```

```
rpcbind   156 root    3u  IPv4 0x300015e96f0      0t0  UDP *:sunrpc (Idle)
rpcbind   156 root    4u  IPv4 0x300015e9470      0t0  UDP *:* (Unbound)
rpcbind   156 root    5u  IPv4 0x300015e91f0      0t0  UDP *:32771 (Idle)
rpcbind   156 root    6u  IPv4 0x300015e8e30      0t0  TCP *:sunrpc (LISTEN)
rpcbind   156 root    7u  IPv4 0x300015e8bb0      0t0  TCP *:* (IDLE)
sshd      242 root    5u  IPv4 0x300016b9bf8      0t0  TCP *:22 (LISTEN)
xntpd     260 root   19u  IPv4 0x300016b9838      0t0  UDP *:ntp (Idle)
xntpd     260 root   20u  IPv4 0x300015e9d30      0t0  UDP localhost:ntp (Idle)
xntpd     260 root   21u  IPv4 0x300016b95b8      0t0  UDP loki:ntp (Idle)
xntpd     260 root   22u  IPv4 0x300015e86b0      0t0  UDP loki-js:ntp (Idle)
mountd    277 root    6u  IPv4 0x300016b8e38      0t0  UDP *:32782 (Idle)
mountd    277 root    8u  IPv4 0x300016b8938      0t0  TCP *:32771 (LISTEN)
nfsd      279 root    4u  IPv4 0x300016b8cf8      0t0  UDP *:nfsd (Idle)
nfsd      279 root    5u  IPv4 0x300016b8578      0t0  TCP *:nfsd (LISTEN)
rpc.bootp 284 root    0u  IPv4 0x300016b8078      0t0  UDP *:32783 (Idle)
rpc.bootp 284 root    1u  IPv4 0x300018c1c00      0t0  TCP *:32772 (LISTEN)
```

**Can't Write In /usr**

Verify /usr is mounted are read only.

```
# touch /usr/bin/bad
touch: /usr/bin/bad cannot create
```

**Can't Run Set-UID From /var/tmp**

Verify that executables can't be run setuid.

```
# cd /var/tmp
# cp /usr/bin/su .
# exit
$ /usr/bin/su
Password:
# exit
$ /var/tmp/su
Password:
su: Sorry
```

**IP Filter rules**

With the configuration created when IP Filter was first installed, only ssh from 10.1.1.0/24 is permitted on the "public" interface. The jumpstart interface (hme1) only allows broadcast traffic by default. Any time addclient is ran, the appropriate rules will be installed.

```
# ipfstat -i
block in log quick from any to any with short
block in log quick from any to any with ipopt
```

```
pass in quick on hme0 proto tcp from 10.1.1.0/24 to 10.1.1.1/32 port = 22
pass in quick on hme1 from 192.168.200.0/24 to 255.255.255.255/32
pass in quick on hme1 from 192.168.200.0/24 to 192.168.200.255/32
block in log from any to any
```

**Check the logs**

Verify output in the log files.
/var/adm/messages

```
Dec 23 01:19:19 blah su: [ID 810491 auth.crit] 'su root' failed for huffner on /dev/console
Dec 23 01:19:31 blah genunix: [ID 809163 kern.info] NOTICE: su, uid 100: setuid \
        execution not allowed, dev=200000007c
Dec 23 01:19:34 blah su: [ID 810491 auth.crit] 'su root' failed for huffner on /dev/console
```

/var/log/authlog

```
Dec 23 01:00:22 blah sshd[483]: [ID 947420 auth.warning] refused connect from \
        192.168.200.1
Dec 23 01:07:52 blah sshd[503]: [ID 800047 auth.info] Failed password for root \
        from 192.168.200.1 port 32810 ssh2
Dec 23 01:07:54 blah sshd[503]: [ID 800047 auth.info] Connection closed by \
        192.168.200.1
Dec 23 01:08:17 blah sshd[504]: [ID 800047 auth.info] Failed password for root \
        from 192.168.200.1 port 32811 ssh2
```

**Verifying Jumpstart**

Apply the testing and verification process used in Solaris Practicum on a client that has just been jumpstarted. This is the same group of tests in the previous section.

**Can get here from there**

Testing the ability to ssh to the new system.

```
huffner@loki> ssh blah
The authenticity of host 'blah (192.168.200.66)' can't be established.
RSA key fingerprint is 23:89:7b:b9:cb:25:79:a1:d0:ba:e6:ba:57:d7:7e:a5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'blah,192.168.200.66' (RSA) to the list of known hosts.
huffner@blah's password:
##########################################################################
```

```
###
…Message of the day omitted…
huffner@blah>
```

**Can't Get Here From There**

Testing the TCP Wrappers denial of service from disallowed system.

```
huffner@badhost> ssh blah
ssh_exchange_identification: Connection closed by remote host
```

On the remote server the mail to root shows:

```
From: Super-User <root>
To: root@yourdomain.com
Subject: sshd@blah: connection attempt from 192.168.200.1
```

**Can't Get Here As Superuser**

Testing ssh to host as root.

```
# ssh blah
The authenticity of host 'blah (192.168.200.66)' can't be established.
RSA key fingerprint is 23:89:7b:b9:cb:25:79:a1:d0:ba:e6:ba:57:d7:7e:a5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'blah,192.168.200.66' (RSA) to the list of known hosts.
root@blah's password:
Permission denied, please try again.
root@blah's password:
Permission denied, please try again.
root@blah's password:
Permission denied (publickey,password,keyboard-interactive).
```

**What's Running**

Listing of what is running and what ports are listening.

```
# ps -ef
     UID   PID  PPID  C    STIME TTY      TIME CMD
    root     0     0  0 22:00:52 ?        0:00 sched
    root     1     0  0 22:00:52 ?        0:00 /etc/init -
    root     2     0  0 22:00:52 ?        0:00 pageout
    root     3     0  0 22:00:52 ?        0:05 fsflush
    root   280     1  0 22:02:00 console  0:00 ksh -o vi
    root   242     1  0 22:01:20 ?        0:00 /usr/lib/inet/xntpd
    root   112     1  0 22:01:15 ?        0:00 ipmon -s
    root    51     1  0 22:00:56 ?        0:00 /usr/lib/sysevent/syseventd
```

```
   root    53     1  0 22:00:56 ?        0:00 /usr/lib/sysevent/syseventconfd
   root   162     1  0 22:01:18 ?        0:00 /usr/sbin/cron
   root   282     1  0 22:02:20 ?        0:00 /usr/lib/sendmail -q15m
   root   213     1  0 22:01:19 ?        0:00 /usr/lib/utmpd
   root   256   242  0 22:01:21 ?        0:00 /usr/lib/inet/xntpd
   root   235     1  0 22:01:20 ?        0:01 /usr/local/bin/snort -i hme0 -d -o -D -c
/etc/opt/snort/snort.conf
   root   273     1  0 22:01:21 ?        0:01 /usr/local/sbin/sshd
   root   415     1  0 00:09:01 ?        0:00 /usr/sbin/syslogd -t
   root   507   280  0 01:10:27 console  0:00 ps -ef
# lsof -i
COMMAND PID USER   FD   TYPE        DEVICE SIZE/OFF NODE NAME
xntpd   242 root   19u  IPv4 0x30000d62ba8      0t0  UDP *:ntp (Idle)
xntpd   242 root   20u  IPv4 0x30000d63e68      0t0  UDP localhost:ntp (Idle)
xntpd   242 root   21u  IPv4 0x30000d595b0      0t0  UDP blah:ntp (Idle)
sshd    273 root    5u  IPv4 0x30000d63d28      0t0  TCP *:22 (LISTEN)
syslogd 415 root    3u  IPv4 0x30000d77960      0t0  UDP *:syslog (Idle)
```

**Can't Write In /usr**

Verify /usr is mounted are read only.

```
# touch /usr/bin/bad
touch: /usr/bin/bad cannot create
```

**Can't Run Set-UID From /var/tmp**

Verify that executables can't be run setuid.

```
# cd /var/tmp
# cp /usr/bin/su .
# exit
$ /usr/bin/su
Password:
# exit
$ /var/tmp/su
Password:
su: Sorry
```

**IP Filter rules**

After a jumpstart, IP Filter will block everything by default. Once an administrator makes changes to the /etc/opt/ipf/ipf.conf, no one will have access to the system. (This is due the entry "hostname" in the default ipf.conf copied over during the install.)

```
# ipfstat -i
pass in quick on lo0 from any to any
```

```
block in log quick from any to any with short
block in log quick from any to any with ipopt
block in log quick from any to any
```

**Check the logs**

Verify output in the log files.
/var/adm/messages

```
Dec 23 01:19:19 blah su: [ID 810491 auth.crit] 'su root' failed for huffner on /dev/console
Dec 23 01:19:31 blah genunix: [ID 809163 kern.info] NOTICE: su, uid 100: setuid \
        execution not allowed, dev=200000007c
Dec 23 01:19:34 blah su: [ID 810491 auth.crit] 'su root' failed for huffner on /dev/console
```

/var/log/authlog

```
Dec 23 01:00:22 blah sshd[483]: [ID 947420 auth.warning] refused connect from \
        192.168.200.1
Dec 23 01:07:52 blah sshd[503]: [ID 800047 auth.info] Failed password for root \
        from 192.168.200.1 port 32810 ssh2
Dec 23 01:07:54 blah sshd[503]: [ID 800047 auth.info] Connection closed by \
        192.168.200.1
Dec 23 01:08:17 blah sshd[504]: [ID 800047 auth.info] Failed password for root \
        from 192.168.200.1 port 32811 ssh2
```

This system, which has been jumpstarted, passes all the tests that were in the Solaris Practicum.

**Vulnerability Check**

With all the tools in place, it is now time to test the vulnerabilities discussed in the Risk Analysis section.

**Network Access**

As stated in the Risk Analysis, TFTP, RPC, and NFS are all notoriously insecure, yet the services are required by Jumpstart. To protect the Jumpstart server, IP Filter is installed. This way, only desired systems have access to the necessary services. For this setup, the SSH is the only traffic allowed on the interface connected to the LAN. The other interface is connected to either another system via cross-over cable or to a switch that is not routed. Then only RPC, NFS, and TFTP are permitted, and only at a requested times. The administrator has to run the addclient script to open the services for a jumpstart client and the services will only be open for four hours. However, because RPC is difficult to firewall, everything is opened up the client's IP address. That is the reason for the second network and for keeping it completely private. Using **nmap**, here is a scan of the system on the corporate LAN without IP Filter enabled. The first option tells nmap to perform a TCP scan, the second tells it to perform a scan of UDP services, and the final option tells nmap not to ping the host.

```
$ sudo nmap -sT -sU -P0 loki
Password:

Starting nmap V. 2.54BETA22 ( www.insecure.org/nmap/ )
Interesting ports on  (10.1.1.1):
(The 3116 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
69/udp     open        tftp
111/tcp    open        sunrpc
111/udp    open        sunrpc
123/udp    open        ntp
2049/tcp   open        nfs
2049/udp   open        nfs
32771/tcp  open        sometimes-rpc5
32771/udp  open        sometimes-rpc6
32772/tcp  open        sometimes-rpc7


Nmap run completed -- 1 IP address (1 host up) scanned in 561 seconds
```

Notice the large number of ports listening. Turning on filtering gives very different results:

```
$ sudo nmap -sT -sU -P0 loki
Password:

Starting nmap V. 2.54BETA22 ( www.insecure.org/nmap/ )
Interesting ports on  (65.18.2.232):
(The 3125 ports scanned but not shown below are in state: filtered)
Port       State       Service
22/tcp     open        ssh


Nmap run completed -- 1 IP address (1 host up) scanned in 2050 seconds
```

As configured, only ssh is listening on port 22. Now if a scan were to be run on the private interface without the filtering enabled, the same ports would be open as on the first scan on the primary interface. Running nmap with filtering enabled shows as planned, no ports are listening. Only TCP ports were scanned due to the amount of time UDP scans can take.

```
# nmap -sT -P0 192.168.200.1

Starting nmap V. 2.54BETA22 ( www.insecure.org/nmap/ )
All 1523 scanned ports on  (192.168.200.1) are: filtered

Nmap run completed -- 1 IP address (1 host up) scanned in 1712 seconds
```

Now when addclient is run, the filtering rules are opened up for that IP address and the scan looks almost exactly the same as the first scan. Note that UDP ports were not scanned. This is because a scan of both UDP and TCP was done and it had not completed in over two hours

```
# nmap -sT -P0 192.168.200.1
Starting nmap V. 2.54BETA22 ( www.insecure.org/nmap/ )
Interesting ports on  (192.168.200.1):
(The 1518 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
111/tcp    open        sunrpc
2049/tcp   open        nfs
32771/tcp  open        sometimes-rpc5
32772/tcp  open        sometimes-rpc7


Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

Here a limited scan of UDP ports running RPC services is done. UDP will still be listening on ports 69 (TFTP), 111 (RPCBind), and 2049 (NFS).

```
 # nmap -sU -sR -p 32770-32789 -P0 192.168.200.1

Starting nmap V. 2.54BETA22 ( www.insecure.org/nmap/ )
Unable to find nmap-rpc!  Resorting to /etc/rpc
Interesting ports on  (192.168.200.1):
(The 17 ports scanned but not shown below are in state: closed)
Port       State       Service (RPC)
32771/udp  open        sometimes-rpc6
32782/udp  open        unknown
32783/udp  open        unknown
```

As long as the "Jumpstart Network" stays private and physically secure, then there will be very little risk of this system being hacked.

**Trojan horse**

As mentioned in Risk Analysis section, if an attacker puts a Trojan horse in the jumpstart configuration, the attacker would then have control of every jumpstarted system since placement of that Trojan horse. Tripwire is used to detect this attack. When the Tripwire package is installed, a cron job is set to run every night. Below is an example of an attacker placing a Trojan horse on the jumpstart server. This test is accomplished by moving the original OpenSSH.pkg to /opt/jumpstart and creating an empty file as a replacement.

```
# cd /opt/jumpstart/packages
# mv OpenSSH.pkg ..
# mkfile 5m OpenSSH.pkg
# /usr/local/bin/tw/tripwire -c /etc/opt/tw/tw.config
Tripwire(tm) ASR (Academic Source Release) 1.3.1
File Integrity Assessment Software
… output omitted …
font size change for readability…
### Phase 4:   Searching for inconsistencies
###
###                 Total files scanned:          58226
###                       Files added:          1
###                       Files deleted:         0
###                       Files changed:          1
###
###                 Total file violations:         2
###
added:   -rw------- root      4939776 Dec 23 00:26:07 2001 /opt/jumpstart/OpenSSH.pkg
changed: -rw------T root      5242880 Dec 23 00:26:20 2001 /opt/jumpstart/packages/OpenSSH.pkg
### Phase 5:   Generating observed/expected pairs for changed files
###
### Attr       Observed (what it is)        Expected (what it should be)
### =========== ============================ ============================
/opt/jumpstart/packages/OpenSSH.pkg
      st_mode: 101600                    100600
       st_gid: 1                        0
      st_size: 5242880                  4939776
     st_mtime: Sun Dec 23 00:26:20 2001     Wed Dec 19 01:37:08 2001
     st_ctime: Sun Dec 23 00:26:20 2001     Wed Dec 19 01:37:08 2001
   md5 (sig1): 1VDZuEMAbV1ilfkyPYnT:s       1wnUX7W1jJInatHXP:5uyu
 snefru (sig2): 3IolthVKipMWiNa1nxMpme       01SaqWHhgiYqEcBzy8x:Zu
```

Notice the two file changes. If any unexpected changes are found, immediately check with the other administrators to see if any changes were made. If so, update the integrity database with:

```
# tripwire –update /opt/jumpstart/packages/OpenSSH.pkg
```

If the change was not made by a system administrator, immediately take the system off the network and start the analysis.

## Conclusion

Using Jumpstart has many benefits: an administrator spends less time configuring each new install, each system is configured exactly the same, and there is little room for human error. However, Jumpstart does need insecure services to run. Using the steps outlined in this paper, an administrator can properly configure and secure the Jumpstart server.

## Appendix A: Jumpstart Profiles

Server profile with a disk 9GB or larger and no need for running X-based applications:

```
# This install is for servers without a need to remotely run X-based apps.
# This is for root disks => 9GB
#
# Install type
install_type     initial_install
system_type    standalone

# Cluster
cluster  SUNWCreq

# additional packages
package SUNWadmc  add
package SUNWadmfw            add
package SUNWntpr    add
package SUNWntpu   add
package SUNWaccr    add
package SUNWaccu   add
package SUNWzlib     add
package SUNWzlibx   add
package SUNWgzip    add
package SUNWter      add

# Disk layout
root_device c0t0d0s0
partitioning explicit
filesys rootdisk.s0 265 / logging
filesys rootdisk.s1 auto swap
filesys rootdisk.s3 1024 /usr ro,logging
filesys rootdisk.s4 4096 /var logging,nosuid
filesys rootdisk.s7 10 unnamed
filesys rootdisk.s5 free /opt logging,nosuid
```

Server profile with a disk smaller than 9GB and no need for running X-based applications.

```
# This install is for servers without a need to remotely run X-based apps.
# This is for root disks of 4GB
#
# Install type
install_type     initial_install
system_type    standalone

# Cluster
cluster  SUNWCreq

# additional packages
package SUNWadmc  add
package SUNWadmfw           add
package SUNWntpr    add
package SUNWntpu   add
package SUNWaccr    add
package SUNWaccu   add
package SUNWzlib     add
package SUNWzlibx   add
package SUNWgzip    add
package SUNWter      add

# Disk layout
root_device c0t0d0s0
partitioning explicit
filesys rootdisk.s0 265 / logging
filesys rootdisk.s1 auto swap
filesys rootdisk.s3 512 /usr ro,logging
filesys rootdisk.s4 2048 /var logging,nosuid
filesys rootdisk.s7 10 unnamed
filesys rootdisk.s5 free /opt logging,nosuid
```

Server profile with a disk 9GB or larger and a need to run X-based applications from a remote system.

```
# This install is for servers with a need to remotely run X-based apps.
# This is for root disks => 9GB
#
# Install type
install_type     initial_install
system_type      standalone

# Cluster
cluster SUNWCreq

# additional packages
package SUNWadmc  add
package SUNWadmfw          add
package SUNWntpr    add
package SUNWntpu    add
package SUNWaccr    add
package SUNWaccu    add
package SUNWzlib    add
package SUNWzlibx   add
package SUNWgzip    add
package SUNWter     add

# x stuff
package          SUNWxwrtl   add
package          SUNWxwfnt   add
package          SUNWxwice   add
package          SUNWtltk    add
package          SUNWxilrl   add
package          SUNWxildh   add
package          SUNWxilow   add
package          SUNWxwplt   add
package          SUNWmfrun   add

# Disk layout
root_device c0t0d0s0
partitioning explicit
filesys rootdisk.s0 265 / logging
filesys rootdisk.s1 auto swap
filesys rootdisk.s3 1024 /usr ro,logging
filesys rootdisk.s4 4096 /var logging,nosuid
filesys rootdisk.s7 10 unnamed
filesys rootdisk.s5 free /opt logging,nosuid
```

Server profile with a disk smaller than 9GB and a need to remotely run X-based applications.

```
# This install is for servers with a need to remotely run X-based apps.
# This is for root disks of 4GB
#
# Install type
install_type     initial_install
system_type      standalone

# Cluster
cluster  SUNWCreq

# additional packages
package SUNWadmc  add
package SUNWadmfw          add
package SUNWntpr    add
package SUNWntpu    add
package SUNWaccr    add
package SUNWaccu    add
package SUNWzlib    add
package SUNWzlibx   add
package SUNWgzip    add
package SUNWter     add

# x stuff
package          SUNWxwrtl    add
package          SUNWxwfnt    add
package          SUNWxwice    add
package          SUNWtltk     add
package          SUNWxilrl    add
package          SUNWxildh    add
package          SUNWxilow    add
package          SUNWxwplt    add
package          SUNWmfrun    add

# Disk layout
root_device c0t0d0s0
partitioning explicit
filesys rootdisk.s0 265 / logging
filesys rootdisk.s1 auto swap
filesys rootdisk.s3 512 /usr ro,logging
filesys rootdisk.s4 2048 /var logging,nosuid
filesys rootdisk.s7 10 unnamed
filesys rootdisk.s5 free /opt logging,nosuid
```

## Appendix B: Files copied to Jumpstart Client

| File | Destination | Description |
|---|---|---|
| S99finalcleanup | /etc/rc2.d/ | Runs at system startup, deletes unneeded users, changes shells for other users. |
| at.allow | /etc/cron.d/ | Users in this file are allowed to schedule or modify at jobs. |
| cron | /etc/default | Sets cron to log several lines of output for each job ran. |
| cron.allow | /etc/cron.d/ | Users permitted to add or modify cron jobs. Other users will still be able to run jobs |
| defaultrouter | /etc/ | Sets default route of system upon boot and turns off dynamic routing protocols. |
| ftpusers | /etc/ | Users not permitted to connect via ftp |
| hosts.allow | /etc/ | TCP Wrappers allow file. Systems in this file are allowed to connect via the specifed method(s). |
| hosts.deny | /etc/ | TCP Wrappers deny file. All entries in here are denied access. Last entry is deny all. |
| inetinit | /etc/default/ | Forces system to use stronger algorithm for TCP sequence numbers |
| init | /etc/default/ | Sets default umask for processes started at boot. |
| inittab | /etc/ | Disables listener on serial ports |
| ipf.conf | /etc/opt/ipf/ | Default firewall rules. |
| issue | /etc/ | If used, prints out same legal information as motd |
| login | | Default user settings. |
| md5 | /usr/local/bin | For verifying |
| motd | /etc/ | Message of the day. Has standard legal text. |
| netconfig | /etc/init.d | Sets network parameters a boot. S69netconfig in rc2.d is a link to file in /etc/init.d |
| newinetsvc | /etc/init.d | When the system is booted, inetd starts up with additional logging enables |

| | | |
|---|---|---|
| newsyslog | /etc/init.d | Syslogd starts up in non-listening mode. |
| nsswitch.conf | /etc/ | Name service file. |
| ntp.conf | /etc/inet | NTP configuration file. |
| pam.conf | /etc/ | Disallows logins via rhosts files |
| passwd | /etc/default/ | Enables password aging and sets minimum length to six. |
| perf | /etc/init.d/ | Starts performance monitoring whenever system boots |
| resolv.conf | /etc/ | Contains DNS server information. |
| rotatelog | /usr/local/bin | Rotates logs, cron entries are also added for root |
| sa1 | /usr/lib/ | Modified to have 30 days of performance logs instead of seven. |
| sa2 | /usr/lib/ | Modified to have 30 days of performance logs instead of seven. |
| sendmail | /etc/init.d | Sendmail startup script. This takes |
| sendmail.cf | /etc/mail/ | Sendmail configuration file. |
| sshd_config | /usr/local/etc/ | Configuration file for sshd |
| su | /etc/default/ | Configuration for when users su to other users |
| sys.crontab | /var/spool/cron/crontabs/sys | User sys crontab entry. Used for performance monitoring. |
| syslog.conf | /etc/ | Syslog configuration file, tells syslog what and where to log. |
| System | /etc/ | System parameters set at startup. |

## End Notes

## References

Acheson, Steve, John Green, and Hal Pomeranz. Topics in UNIX Security. SANS Institute, 2001.

Australian Computer Emergency Response Team, CERT Coordination Center. "UNIX Security Checklist v2.0." 8 October 2001. URL: http://www.cert.org/tech_tips/usc20.html#2.17 (23 November 2001).

Brotzman, Lee and Hal Pomeranz. Linux/Solaris Practicum. SANS Institute, 2001.

Chistensen, Steven M. "Creating pkgadd Software Packages Under Solaris." 5 July 2001. URL: http://www.sunfreeware.com/pkgadd.html. (23 November 2001).

Dibowitz, Phil. "IPFilter FAQ." 3 December 2001. URL: http://coombs.anu.edu.au/~avalon/faq/IPFtoc.html (30 November 2001).

Howard, John S. and Alex Noordergraff. JumpStart Technology. Palo Alto: Sun Microsystems Press, 2001.

Pomeranz, Hal. Common Issues and Vulnerabilities in UNIX Security. SANS Institute, 2001.

Pomeranz, Hal. UNIX Security Tools. SANS Institute, 2001.

Sun Microsystems Incorporated. "Solstice Disksuite 4.2.1 Reference Guide." 2000. URL: http://docs.sun.com/ab2/coll.260.2/DISKSUITEREF/@Ab2TocView?Ab2Lang=C&Ab2Enc=iso-8859-1. (19 November 2001).

Tripwire Academic Source Release. Tripwire Security Systems Incorportated, 1999.

Woehr, Jack. "Building 64-bit GCC 3.0." v 1.5 3 October 2001. URL: http://www.well.com/~jax/rcfb/solaris_tips/build_gcc_3.0_64bit.html (23 November 2001).

[1] Pomeranz,, p. 163.
[2] Australian Computer Emergency Response Team, CERT Coordination Center. "UNIX Security Checklist v2.0." Oct 8, 2001. URL: http://www.cert.org/tech_tips/usc20.html#2.17 (23 November 2001).
[3] Brotzman and Pomeranz, p.133.
[4] Brotzman and Pomeranz, pp.134-138.
[5] Danielyan, pp. 134-135.
[6] Brotzman and Pomeranz, p.140.
[7] Brotzman and Pomeranz, p.141.
[8] Brotzman and Pomeranz, p.142.

[9] Brotzman and Pomeranz, p.143.
[10] Brotzman and Pomeranz, p.146.
[11] Brotzman and Pomeranz, p.152.
[12] Brotzman and Pomeranz, p.170.
[13] Brotzman and Pomeranz, p.171.
[14] Brotzman and Pomeranz, p.172.
[15] Brotzman and Pomeranz, p.172.
[16] Brotzman and Pomeranz, p.173.
[17] Brotzman and Pomeranz, p.175.
[18] Brotzman and Pomeranz, p.176.
[19] Brotzman and Pomeranz, p.177.
[20] Brotzman and Pomeranz, p.180.
[21] Brotzman and Pomeranz, pp. 182-183.
[22] Brotzman and Pomeranz, pp. 184-185.
[23] Brotzman and Pomeranz, pp. 186-188.
[24] Brotzman and Pomeranz, pp. 191-194.
[25] Brotzman and Pomeranz, pp. 195-197.
[26] Brotzman and Pomeranz, p. 198.
[27] Woehr, Jack. "Building 64-bit GCC 3.0." v 1.5 3 October 2001. URL: http://www.well.com/~jax/rcfb/solaris_tips/build_gcc_3.0_64bit.html (11 November 2001).
[28] Brotzman and Pomeranz, p. 163.
[29] Brotzman and Pomeranz, p. 161.
[30] Tripwire Security Systems Incorporated. Tripwire Academic Source Release 1.3.1 for UNIX User Manual. (30 April 1999). p 8.
[31] Pomeranz. Common Issues and Vulnerabilities in UNIX Security. p. 143.
[32] Pomeranz. UNIX Security Tools. p. 46.
[33] Pomeranz. UNIX Security Tools. p. 47.
[34] Brotzman and Pomeranz, pp. 199-203.
[35] Howard and Noordergraff, pp 9-13.
[36] Howard and Noordergraff, pp.13-15.
[37] Howard and Noordergraff, p. 17.
[38] Howard and Noordergraff, pp. 40-46.
[39] Howard and Noordergraff, p. 50.
[40] Howard and Noordergraff, p. 50.
[41] Howard and Noordergraff, p. 61-63.
[42] Howard and Noordergraff, p. 59.
[43] Howard and Noordergraff, p. 39.
[44] Brotzman and Pomeranz, p. 203-211.