Global Information Assurance Certification Paper

SANS GIAC Certification

Level 2 GCUX
Certified Unix Security Administrator
Practical Assignment v 1.8
Installing a Secure Solaris 8 Nameserver

Submitted by
Deanne Palmer
February, 2002

Contents

Introduction

The process of installing a new system on any network can be fraught with security risks. 'rootkits' and 'buffer overflows' refer to a few popular methods of hacking into a unsecured system. When the system is a nameserver, 'poison caching', 'data mining' and 'session hijacking' can be serious additional risks. We will attempt to document steps to reduce the possibility of an intrusion before, during and after the new operating system installation for a new nameserver for our network.
The vulnerabilities of a networked system can be greatly reduced by a secure configuration. The default settings from the vendor typically implement all the bells and whistles they have been advertising, but not necessarily the most secure environment possible. By changing the defaults and securing the system before it is deployed on the network, the security of the system and the network can be greatly improved. The installation process is just the first step. Upgrading, and monitoring are also critical to keeping a secure system.

Description of the System

This paper will attempt to document the steps necessary for the implemention of a secure external DNS server on a Solaris 8 system in a "split horizon" DNS environment. BIND 9.2.0, the most current version of the software available from www.isc.org, will be installed for this purpose. By implementing a "split horizon" DNS we expect to minimize the local host information that is publicly advertised to the Internet, and thereby somewhat reduce our vulnerability to exploit.

Several DNS vulnerabilities have been exploited in recent years. In the CERT advisory 'CA-2001-02', four vulnerabilities in BIND were disclosed. "Three of the vulnerabilities involve buffer overflow conditions that could allow a malicious attacker to execute code as superuser. The fourth vulnerability involves an input validation error that could allow a malicious attacker to read information from the program stack." (http://www.ciac.org/ciac/bulletins/l-030.shtml). According to CERT, buffer overflow vulnerabilities in the TSIG (transaction signature code) and nslookup have been corrected through vendor patches (Jan. 2001) and through the installation of the more recent versions of the software - BIND4.9 or BIND 8.2.3 and higher.

To minimize the impact of a possible exploitation of naming services, CERT also advises that a "split horizon" DNS be implemented. In a "split horizon" DNS scenario, a public DNS server containing only public host names is located on the perimeter network where it will be the advertised DNS server for the enterprise. Another internal DNS server would contain all the local internal host names. The internal DNS server would be located inside the secure internal network. It is possible to implement security policies so that if one of the DNS servers is compromised, the other will still function normally.

Environment

The local network is comprised of an Internet gateway router which connects to a perimeter firewall. Access from the Internet is directed to a screened DMZ network where the external DNS server, an external webserver, and an external mail proxy server are located. An internal firewall allows only specific services to connect from the screened DMZ network to a secure network where the internal DNS, mail hub, an internal webserver, database servers, and local user workstations are located.

Hardware

The hardware for the external DNS server is a SUN Enterprise 250 with two 400 MHz Sparc CPU's and 2048 Mb of memory running Solaris 8 (Sparc v7.01) with two 18 Gb hot-pluggable disks, a SCSI CD ROM, a 100 Mb Ethernet card, and a SCSI 8 mm Mammoth tape drive. The system was packaged with a PGX32 video adapter and color monitor for the console. The server will not be connected to the network until steps to secure the server have been completed. This will prevent the possibility of a system compromise by an intruder during the installation process.
All equipment including serial nos. has been entered in the IT inventory system.

Physical Security

The physical security of the hardware is a critical component to the overall security. Access to the console can provide a prime opportunity for a hacker to get into a system. As noted in the *SANS Institute Track 6 Common Issues and Vulnerabilities in UNIX Security,* physical access can mean 'root' access. If an attacker repeatedly crashes the machine by power-cycling, or repeatedly disconnects and reconnects the console keyboard, the root file system can become corrupted, and require a manual *fsck* . In this case, a root shell will come up and the intruder now has root access. Other means of attack when physical access is available include booting the system from the OS media (Solaris CD), and mounting the root file system, then the attacker can change the root password, or create a set-UID shell.

The system we will be installing is located in a locked and monitored central server room. Electrical power to the room is monitored by a UPS with 30 minute backup capacity and the ability to automatically switch to a gas generator after a 5 minute power failure. Key access to the server room is limited to IT staff. UPS monitoring includes mail and pager notification of status changes.
A dry CO2 fire suppression system has been installed in the server room with an emergency stop button and rotary phone located just outside the entrance. All IT personnel must attend training regarding the operation of the fire suppression system. Documentation is also available.
Backup media are removed from the server room on a daily basis and stored in a remote building. A monthly sample restore of media data is done to assure their validity.

Solaris 8 Installation

It is important to use the most recent version of the operating system, since this will provide the most recent updates of the OS. Together with the latest patch cluster, this can help reduce our vulnerabilities.
The following minimum information is necessary for the installation: the hostname, the IP address, and the netmask of the system to be installed.
Installation process:

1. Power on the console and peripherals then the server (in that order).
2. If the server attempts to boot, enter [STOP]-A to interrupt the boot process. This should bring you to an ok> prompt.
3. Insert the Sparc Solaris 8 Software CD-ROM 1 of 2.
4. At the "ok" prompt, enter "boot cdrom". This starts the installation process.
   ok> **boot cdrom**
5. Select a Language ('0' for English) and Locale (0 English C-7bit ASCII), then select 'Continue'.
6. At prompt "Is system connected to a network?", answer **Yes**.
7. Enter 'No' when prompted regarding DHCP configuration. This server requires a static IP address and DHCP is not recommended on an Internet server.
8. Enter hostname when prompted. The hostname should be no more than 8 characters and must be unique within the host's domain.
   Hostname: shade
9. Select the primary network interface and enter the assigned IP address when prompted.
   Example - IP: 129.200.9.1
10. The next prompt will ask if the 'system is part of a subnet'.
    Answer appropriately (yes/no).
11. If you answer 'yes' to subnetting, you will be prompted to enter the subnet mask. If you do not know the netmask for your subnet, contact your Network Administrator.
    Example - netmask: 255.255.0.0
12. Enter 'No' when prompted regarding enabling Ipv6 on this machine. (We are not using Ipv6 and can enable it later, if necessary.)
13. The installation will present your selections:
    Example:
    Networked: yes
    Use DHCP: no
    Host name: shade
    IP address: 129.200.9.1
    Enable IPv6: no
    System part of subnet: yes
    Netmask: 255.255.255.0

Either 'Continue' if the information is correct, or 'Change' to correct any information.

14. Enter 'No' when prompted to configure Kerberos security on this machine, and 'Confirm'.
15. Enter 'None' when prompted regarding name service. We will enable 'dns' later.
16. Select the appropriate time zone ('US/ Eastern'). Then 'Continue'.
17. Set the date and time if they are not correct. Then 'Continue'.
18. Information regarding the timezone, date and time will be presented.

    Time Zone: US/Eastern
    Date and Time: 01/21/02 15:20:00

    Enter 'Continue' if they are correct; enter 'Change' to correction the information.
19. Select 'Initial' installation when prompted then 'Continue'. This is not an 'upgrade'. System disks will be overwritten by the new Solaris software when an 'initial' installation is selected.
20. When prompted for "geographic regions for which support should be installed", select only "North America – U.S.A. (en US.IS)8859-1)", then 'Continue'.
22. When presented with the selection of Solaris software bundle to install, select the 'Core System Support' software bundle. Sans Institute's *Solaris Security Step By Step* recommends the installation of the smallest operating system image necessary because of the large variety of programs in the other selections. Each program slightly increases the security risks on the system. 'Core system support' will meet most of our needs, but we will later need to install several additional Solaris software packages for this server's function. Select 'Continue'.
23. A list of all the hard drives will be displayed. The boot disk will be listed under 'Selected Disks'. Be sure the 'Required' value is less than the 'Total Selected' value. Add disks by clicking on an 'Available' disk and then on the ">>" button to move it to 'Selected Disks'.
24. Select 'Continue' when prompted to preserve existing disk data. This is a new system, so we have nothing to preserve.
    We could select to preserve data on disks other than the system partitions (/, /usr, /var) if necessary.
25. Select 'Manual Layout' when prompted for "auto-layout to automatically layout file systems". Then select 'Customize' to layout the disk partitions.

Customizing Disks:

A typical disk layout for Core installation on an 18 Gb disk (guideline only)

Slice	Name	Size
0	/	256 Mb
1	Swap	2048 Mb
2	Overlap	16886 Mb
3		
4	/var	2048 Mb
5	/usr	1024 Mb
6	/opt	2048 Mb

7	/usr/local	9348 Mb (remaining space)

A 'Recommended Minimum' size for system partitions is displayed in the right corner of the disk layout screen. This is a guide only. Depending upon the function of the server, various disk partition layouts may be more appropriate. The installation will display an error if the total size of the partitions (excluding overlap) exceeds the total size of the disk. Due to rounding, you may need to reduce the size of one of the partitions incrementally until you are within the calculated disk size.

26. When specifications for the disk layout are completed, select 'OK'.
27. The newly specified file disk layout will be displayed. Select 'Continue' if the information is correct, otherwise go back and correct the layout.
28. Select 'No' when prompted regarding mounting of remote file systems, then 'Continue'.
29. The installation profile is displayed including the disk layout. If this looks correct, select 'Begin Installation'.
30. At the beginning of the installation process, the following message will appear: "After Solaris software is installed, the system must be rebooted. You can choose to have the system automatically reboot, or you can choose to manually reboot the system if you want to run scripts or do other customization before the reboot. You can manually reboot a system using the reboot(1M) command." Select 'Manual Reboot'. This will allow you to watch for any errors as the system reboots.

The install process will format the disk partitions and then install the selected software packages from the CD-ROM. This may take several hours, depending on the system hardware.
After the installation finishes a message indicating the installation is complete and the system should be rebooted will be displayed on the console.

Reboot the server and watch for any errors on the startup.

Root Password

A login prompt will be displayed as the system comes up on the reboot. Login as root and set the password (it is null at this point) using the 'password' command.

```
# passwd
passwd: Changing password for root
New password: ********      (password characters will not display)
Re-enter new password: ********
Passwd (SYSTEM): passwd successfully changed for root
```

Choose a password that follows secure computing standards. It should be a minimum of 8 characters, including upper and lower case letters as well as number and special characters. It should not be easily guessed, a dictionary word, or a proper name.
This password should be included in a list of passwords of network servers that is stored in a secure file cabinet and that is only accessible by selected IT staff in extreme circumstances.

Additional Software Packages

Additional packages beyond the Core software bundle will be needed for this system. Network time protocol, accounting packages and several other packages needed for the installation of other open-source software will be installed.
The software is located on the *Solaris 8 Software 1 of 2* and *Solaris 8 Software 2 of 2* CD's.

1. Insert the Solaris 8 Software 1 of 2 CD.
2. Mount the CD-Rom device (assuming c0t6d0s0 is the CD-Rom)
   /usr/sbin/mount –F hsfs –o ro /dev/dsk/c0t6d0s0 /mnt
   /usr/bin/cd /mnt/Solaris_8/Product
3. Install the packages:
   /usr/sbin/pkgadd –d . <packagename>
   Add the following packages:

SUNWntpr, SUNWntpu	NTPnetwork time protocol
SUNWdoc	manual pages
SUNWbzip	needed for SUNWgzip
SUNWadmc	core system admin libraries
SUNWlibC	compiler bundled libC
SUNWlibms	needed for perl
SUNWxwrtl	for X client applications
SUNWxwfnt	
SUNWxwice	
SUNWtltk	
SUNWxilrl	
SUNWxildh	
SUNWxilow	
SUNWxwplt	
SUNWmfrun	
SUNWpl5u	Perl 5.005_03 needed for OpenSSL portion of SSH

   Answer 'Yes' to prompts regarding 'scripts will be executed as super-user'.
4. When complete, unmount the CD
   /usr/bin/cd /
   /usr/sbin/umount /mnt
5. Insert the Solaris 8 Software 2 of 2 CD.
6. Mount the CD-Rom device (assuming c0t6d0s0 is the CD-Rom)
   /usr/sbin/mount –F hsfs –o ro /dev/dsk/c0t6d0s0 /mnt

/usr/bin/cd /mnt/Solaris_8/Product

7. Install the packages:
   /usr/sbin/pkgadd –d . <packagename>
   Add the following packages:

SUNWaccr	accounting
SUNWaccu	accounting
SUNWter	terminfo database
SUNWman	man pages
SUNWbash	bash shell
SUNWzlib	Zip compression library
SUNWgzip	gzip
SUNWast	system and file access security
SUNWsprot	performance monitoring tools
SUNWpl5p	Perl documentation
SUNWpl5m	Perl reference manual pages
SUNWtoo	ldd, truss, and other tools (32-bit)

   Answer 'Yes' to the prompts.

8. When complete, unmount the CD
   /usr/bin/cd /
   /usr/sbin/umount /mnt

Applying Patches

In order to reduce the vulnerabilities of the system, the latest patches should be installed immediately.
The latest Sun Recommended & Security Patch Cluster for Solaris 8 (sparc) will be installed first. After all the software has been installed, we can use the Patchdiag tool to check for any additional recommended patches.

This server is not yet connected to the network, so the patches will need to be retrieved from a secured network server, then moved to this server for installation.

1. On another secure server on the network, download the latest Recommended & Security Patch Cluster for Solaris 8 (sparc) and the checksums from http://sunsolve.sun.com.
2. Be sure you have adequate disk space on the secured server to download the necessary patches and the unzipped files.
3. Check the MD5 checksum of the patch cluster against the downloaded checksum value.
   md5 8_Recommend.zip
   If the output does not match the Checksums information, do NOT continue. It could mean the downloaded patch cluster has been compromised or otherwise corrupted. In this case, delete the current 8_Recommended.zip and checksums file and retrieve new ones.
4. If the checksum is correct, unzip the patch cluster:
   unzip 8_Recommended.zip

This creates the '8_Recommended' patch directory containing the patch cluster.

5. 'tar' the patch directory to tape so that it can be untarred on the new system.
   Example: tar cvf /dev/rmt/0 8_Recommended
6. Remove the tape from the secure server and insert it into the tape drive on the new system. Download the tar file from the tape into a temporary directory with plenty of disk space for the patches.
   cd /var/tmp
   tar xvf /dev/rmt/0
   This will untar the '8_Recommended' directory into /var/tmp.
7. Run the 'install_cluster' from the 8_Recommended directory.
   cd 8_Recommended
   ./install_cluster
   Watch for errors during the patch installation. In general, error codes '2' and '8' can be ignored. These are attempts to install patches that are already installed, or indications that the package to be patched is not installed. The set of error exit codes are listed in the 'installpatch' script of any patch.

The patch error log is located at /var/sadm/install_data/Solaris_8_Recommended_log. Check the log to be sure there were no serious problems.

8. Remove the Recommended patch cluster and tar file from /var/tmp.
   cd /var/tmp
   /bin/rm –r 8_Recommended.*

9. Reboot the system.

Network Configuration

The following steps should be performed before the system is connected to the network – taken from *SANS Institute's* ***Solaris Security Step By Step***:

1. Set */etc/defaultrouter* to the IP of the system's default router:
   /bin/echo "129.200.1.1" > /etc/defaultrouter

2. Create */etc/notrouter* to disable IP forwarding and not start *in.routed*, or *in.rdiscd*. This may prevent an attacker from forging routing updates and rerouting traffic. (SANS Inst Linux/Solaris Practicum p137)
   /bin/touch /etc/notrouter

3. Create /etc/resolv.conf. At this point we will enter the domain, the IP address of the internal DNS nameserver, and IP addresses of additional nameservers provided by our Internet provider. Using the IP instead of the host name is preferred, since host names may be spoofed. We will be entering this host's IP address as the primary nameserver in */etc/resolv.conf* after our installation and configuration of BIND is completed.
   An example of the *resolv.conf* file:

```
domain foo.com
nameserver 129.200.10.5     (the internal DNS server)
nameserver 12.127.16.67     ( Internet nameserver)
nameserver 12.127.17.71       (Internet nameserver)
```

4. Edit the */etc/nsswitch.conf* file, modifying the 'hosts' line to read:

   ```
   hosts:  files  dns
   ```

   This will cause the system to look at the local */etc/inet/hosts* file, then check DNS.
5. Edit the */etc/inet/hosts* file and modify to include the full canonical name and a few other trusted hosts.

   Example:

   ```
   # Internet host table
   #
   127.0.0.1        localhost
   129.200.9.1      shade  loghost  shade.foo.com
   129.200.10.5     intdns intdns.foo.com
   129.200.11.5     mailhost.foo.com mail.foo.com
   ```

Boot Directories

In order to prevent some services from automatically starting at boot time, we will be renaming the links in the various */etc/rc*.d* directories. Many of these services are unnecessary or dangerous and limiting them reduces can reduce our vulnerabilities.

SANS Institute's ***Solaris Security Step by Step*** provides all the details in 'Step 2.1 Purging Boot Directories of Unnecessary Services'. By following these steps, we can prevent the *sysid.net, sysid.sys, autoinstall, nfs.client, nfs.server, rpc, nscd, ldap, preserve* and *sendmail* scripts from executing at start-up. Do this by renaming their links in */etc/rc2.d* and /etc/rc3.d.

```
cd   /etc/rc2.d
mv  S30sysid.net   .s30sysid.net
mv  S71sysid.sys   .s71sysid.sys
mv  S72autoinstall   .s72autoinstall
mv  S73nfs.client   .s73nfs.client
mv  S74autofs    .s74autofs
mv  S73cachefs.daemon    .s73chefs.daemon
mv  S93cacheos.finish   .s93cacheos.finish
mv  S71rpc       .s72rpc
mv  S76nscd      .s76nscd
mv  S71ldap.client   .s71ldap.client
mv  S88sendmail    .s88sendmail
mv  S80PRESERVE    .s80PRESERVE

cd   /etc/rc3.d
mv  S15nfs.server    .s15nfs.server
```

By renaming the links to begin with '.'(dot), they will not be executed at system startup and also will not display on a normal 'ls' list, but they will be available if needed in the future.

Solaris Step by Step 'Step 2.2 New and Modified Boot Services' also provides directions for modifying or disabling certain boot services by creating new scripts in */etc/init.d* and making or recreating the appropriate links in the */etc/rc?.d* directories.

We will not need to modify the default umask since Solaris 8 uses the 'CMASK' parameter in /etc/default/init to control the default umask for system processes.
We will also not be replacing the 'devfsadm' startup script because we have hot-pluggable hardware on this system, and this provides the hot-pluggable hardware support.

Inetd

Inetd provides the listening connection for a variety of network services and is typically initiated at system startup. When a connection is requested on a specific port, inetd translates the port number to a specific service daemon, starts the daemon, and attaches the daemon to the port. TCP wrappers provides easy access control for traditional network server daemons. When inetd receives a request, it passes it to tcpd. Tcpd restricts access to network services based on the source IP of the connection request, through /etc/hosts.allow and /etc/hosts.deny. It first checks hosts.allow. If a match is found, the service is allowed. If no match is found, then it checks hosts.deny. If a match is found, the service is denied, otherwise the service is allowed (by default). The access rules in hosts.allow and hosts.deny contain the service name and the hosts that are allowed/denied access to the service.

In our configuration, we will not be running inetd. We have no need for any internet services on this system. The BIND process will be initiated at startup and does not require any other internet daemons. The safest course would be to remove the inetd configuration file, or at a minimum, comment out all the services in the file(/etc/inetd/inetd.conf).

1. Replace */etc/init.d/inetsvc* with the file */etc/init.d/newinetsvc* from Appendix A of SANS Institute's Solaris Security Step By Step. This disables DHCP, inetd, and multicast routing. We will also remove the name server startup (/usr/sbin/in.named). A separate script will be created later for booting the name service.
   Solaris Security Step By Step */etc/init.d/newinetsvc* script:

```
#!/bin/csh
# newinetsvc  from Solaris Security Step By Step
#
/usr/sbin/ifconfig –au netmask + broadcast +
# Note:  We are removing the name server startup.
```

```
# if [ -f /usr/sbin/in.named  -a  -f  /etc/named.conf ] ; then
#                 /usr/sbin/in.named
#                 echo  "starting internet domain name server."
# fi
#  mcastif=`uname –n`
# echo "Setting default interface for multicast : \c"
# /usr/sbin/route  add –interface –netmask "240.0.0.0" "224.0.0.0"   \
"$mcastif"
# To run inetd in "standalone" mode ( -s flag) uncomment next line
#  /usr/sbin/inetd  -s  -t
# End of script
```

With inetd disabled, we will need to install sshd in order to access this server from the network.

2. Perform Solaris Security Step By Step - Steps 2.2.7 – 2.2.9:
   Copy */etc/init.d/syslog* as */etc/init.d/newsyslog*, then modify *newsyslog* to call syslogd using the *–t* flag like the following:

   ```
   /usr/sbin/syslogd  -t >/dev/msglog 2>&1 &
   ```

   This will stop the syslog daemon from listening on UDP port 514 for messages from other hosts. Set permissions of /etc/init.d/newsyslog to match those of old syslog.
   Remove the old links to syslog from /etc/rc2.d, and create new links to /etc/init.d/newsyslog:

   ```
   ln –s /etc/init.d/newsyslog   S74syslog
   ```

Kernel Configuration

'ndd' provides a method for setting kernel parameters. We will be using it to configure certain network parameters to secure our system.
The following script, /etc/init.d/netconfig, taken from Solaris Security Step By Step should be linked as /etc/rc2.d/S69netconfig.

1. Create the following */etc/init.d/netconfig* script. (Solaris Security Step By Step - Step 2.3.1)

```
#!/sbin/sh
# netconfig script
#
#  Reduce half-open TCP connections in the queue and  the interval allowed
#   to reduce SYN flood vulnerabilities
ndd  -set   /dev/tcp  tcp_conn_req_max_q0   8192
ndd  -set   /dev/tcp   tcp_ip_abort_cinterval  60000

#  Disable response to ICMP timestamp requests and netmask queries
#  to reduce smurf attacks and block mapping vulnerabilities.
ndd  -set   /dev/ip  ip_respond_to_timestamp  0
ndd  -set   /dev/ip  ip_respond_to_timestamp_broadcast  0
```

```
ndd  -set   /dev/ip  ip_respond_to_address_mask_broadcast  0

# Disable ICMP redirects  to prevent denial of service attacks.
ndd  -set   /dev/ip  ip_ignore_redirect  1
ndd  -set   /dev/ip  ip_send_redirects 0

# Disable  source routing.
ndd  -set   /dev/ip  ip_forward_src_routed 0

#  Disable routing of broadcast packets to the local network
ndd  -set   /dev/ip  ip_forward_directed_broadcasts  0

#  Disable IP-forwarding to prevent system being used as a router
ndd  -set   /dev/ip  ip_forwarding 0

#  Turn on strict destination  multi-homing - so can't route from hme0  to hme1(or
#   vice-versa).
ndd  -set   /dev/ip  ip_strict_dst_multihoming  1

#  Reduce the time arp information is available in the arp cache to reduce risk
#  of spoofing attacks.
ndd  -set   /dev/arp   arp_cleanup_interval   60000
ndd  -set   /dev/ip   ip_arp_interval  60000
# end  netconfig script
```

2. Set the ownership/permissions for netconfig and create the link in /etc/rc2.d. (Steps 2.3.2 – 2.3.3)

```
chown   root:root   /etc/init.d/netconfig
chmod   744   /etc/init.d/netconfig
cd    /etc/rc2.d
ln –s  /etc/init.d/netconfig   S69netconfig
```

3. Add the following lines to the bottom of */etc/system* to prevent and log buffer overflow attacks, limit resource consumption, and limit NFS client requests to the privileged port range (from <u>*Solaris Security Step By Step*</u> Steps 2.3.5-2.3.6): (bottom of */etc/system*)

```
* Attempt  to prevent and log stack-smashing attacks
set  noexec_user_stack  =  1
set noexec_user_stack_log  = 1

*  Limit  maximum per user processes
set maxuprc  = 128

 *   Prevent system from creating core files
set sys:coredumpsize = 0
```

```
* Require NFS clients to use privileged ports
set nfssrv:nfs_portmon  = 1
```

4. Reboot to make the changes effective.
```
reboot
```

Housecleaning

Since we will not be using NFS, we will remove the NFS related files, such as */etc/auto_** and */etc/dfs/dfstab*. This should make auditing easier.
Removing the empty 'adm' and 'lp' crontabs files is also recommended.
If any of these files should reappear, something is wrong.
(SANS Institute – 6.5 Linux/Solaris Practicum).
Perform the following commands for this cleanup:

```
/bin/rm   /etc/auto_*    /etc/dfs/dfstab
/bin/rm   /var/spool/cron/crontabs/adm    /var/spool/cron/crontabs/lp
```

File System Options

Security of the files systems can be enhanced by setting certain options in the */etc/vfstab* file. To prevent Trojan horse programs from replacing binaries, */usr* should be set to read-only. */var* and */usr/local* should be set 'nosuid' to prevent rogue set-UID programs from being installed. However, since we will be running a 'chroot'ed environment for BIND, we will not set 'nosuid' on these partitions because it would not function. The 'chroot'ed environment assumes 'suid' to run the application.
The root partition, /, should have the 'logging' option set, to prevent file system inconsistencies that can slow or abort the sytem boot process. Logging turns on a simple transactional file system. This could prevent an attacker, who repeatedly crashes the system, from corrupting the root file system.
According to the man pages for 'mount_ufs', "since the root (/) file system is mounted read-only by the kernel during the boot process, only the 'remount' option (and options that can be used in conjunction with remount) affects the root (/) entry in the /etc/vfstab file." Root, '/', is mounted before /etc/vfstab is read, so it is necessary to use the 'remount' option in /etc/vfstab in order for the 'logging' option to become effective.

Modify the / and */usr* entries in */etc/vfstab* to reflect these changes:

```
/dev/dsk/c0t3d0s0  /dev/rdks/c0t3d0s0    /         ufs      1    no    remount, logging
/dev/dsk/c0t3d0s5  /dev/rdsk/c0t3d0s5    /usr      ufs      1    no    -
/dev/dsk/c0t3d0s4  /dev/rdsk/c0t3d0s4    /var      ufs      1    no    -
/dev/dsk/c0t3d0s6  /dev/rdsk/c0t3d0s6    /usr/local  ufs    2    yes   -
```

We will be changing the options on /usr to be 'read-only'after we have finished our installation of sendmail. We would not be able to modify the /usr/lib/mail files if we set it now and rebooted. A 'reboot' is necessary to make these changes to /etc/vfstab effective.

NTP

NTP(Network Time Protocol) is used to synchronize the clocks on the systems around the world. The 'ntpdate' utility which comes with the Solaris distribution corrects the system clocks based on a variety of options. More information regarding NTP is available from http://www.eecis.udel.edu/~ntp/ntp_spool/html/index.htm. Depending on your requirements for accurate time, you may need to use other options. For our needs, we will be synchronizing once a day with a Public Time Server. Add the following line to the root crontab file:

0 1 * * * /usr/sbin/ntpdate 128.59.64.60 > /dev/null 2>&1

Synchronizing the time across your systems is very important for logging. Time differences can limit your ability to follow the trail of a hacker or potential hacker.

SSH

'telnet' or 'rlogin/rsh' are typically used for accessing the network, but they present numerous security risks. 'ftp' and 'rcp' are used for copying files. The largest security problem with these programs/services is that they send passwords in cleartext over the network.

SSH provides a replacement for 'telnet', the r-commands, and 'ftp'. The encryption provided with SSH can prevent the 'sniffing' of passwords and TCP session hijacking. By using SSH, the inherent insecurity of the r-commands (rlogin, rsh, rcp) can also be removed.

SSH provides host-to-host encryption and allows tunneling of other protocols. Several forms of authentication are supported, including RSA(version 1) and DSA(version 2). It's important to note that OpenSSH combines the SSH1 and SSH2 in the same binary. OpenSSH is free and can be downloaded from http://www.openssh.org or a pre-compiled binary for Solaris is available in package format from http://www.sunfreeware.com. www.sunfreeware.com is a great source of software for Solaris systems. The latest version is OpenSSH-3.0.2.p1. This requires the installation of OpenSSL-9.0.6b (secure socket layer), zlib (zlib compression libraries), and prng (pseudo random number generator). They also recommend installing perl (compiler), egd (entropy gathering daemon), tcp_wrappers, and either gcc(Gnu 'C' compiler package) or libgcc(libraries for gcc). All of this software is freely available for Solaris 8 at the www.sunfreeware.com website. Perl is included on the Solaris 8 distribution and has been installed.

OpenSSH and the related necessary software will be downloaded into a directory on one of our secure servers with Internet access. Then a tar file will be created of that directory onto a tape. The tape will be transferred to our new system and untarred into a temporary /usr/local/tmp directory.

On the secure server, ftp the following packages from www.sunfreeware.com into /tmp/new directory:

zlib-1.1.3-sol8-sparc-local.gz
prngd-0.9.23-sol8-sparc-local.gz

egd-0.8-sol8-sparc-local.gz
libgcc-3_0_3-sol8-sparc-local.gz (gcc libraries required for openssh)
tcp_wrappers-7_6-sol8-sparc-local.gz
openssl-0_9_6b-sol8-sparc-local.gz
openssh-3_0_2p1-sol8-sparc-local.gz
gzip-1.3-sol8-sparc-local

Then create a tar file of /tmp/new onto tape.

```
cd /tmp
tar cf /dev/rmt/0 new
```

Move the tape to the tape drive on the new system:

```
cd /usr/local/
```

Untar the file from the tape into /usr/local/new.

```
tar xvf /dev/rmt/0
```

All the new packages are now located in /usr/local/new:
Install the software.

```
cd /usr/local/new
```

First install the gzip package because you will need gunzip for the others. It will install in /usr/local/bin.

```
pkgadd gzip-1.3-sol8-sparc-local
```

Now unzip and install the other packages.

```
gunzip zlib-1.1.3-sol8-sparc-local.gz
pkgadd –d zlib-1.1.3-sol8-sparc-local
```

The console will display the following:

```
The following packages are available:
 1 SMCzlib     zlib      (sparc) 1.13
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: 1
```

Answer '**y**'es to the prompts.
When it completes, "Installation of <SMCzlib> was successful" will be displayed.

After 'zlib' is installed, follow the same procedure for installing the remaining packages in this specific order, to satisfy their dependencies.

1. prngd-0.9.23-sol8-sparc-local.gz (SMCprngd)
2. egd-0_8-sol8-sparc-local.gz (SMCegd)
3. tcp_wrappers-7_6-sol8-sparc-local.gz (SMCtcpwr)
4. libgcc-3_0_3-sol8-sparc-local.gz (SMClibgcc)

5. openssl-0_9_6b-sol8-sparc-local.gz (SMCossl)

6. openssh-3_0_2p1-sol8-sparc-local.gz (SMCossh3)

Excellent detailed installation and setup instructions of SSH for Solaris are provided in the Sun Blueprints document "Building and Deploying OpenSSH for Solaris Operating Environment" by Jason Reid and Keith Watson at http://www.sun.com/blueprints/0701/openSSH.pdf.

After the package installations are completed, check to be sure /usr/local/bin and /usr/local/sbin are in your PATH environment variable, and be sure /usr/local/bin is in your path first.

To set the PATH variable -

for sh/ksh:

```
PATH =/usr/local/bin:/usr/local/sbin:$PATH
export $PATH
```

for csh:

```
set path  = (  /usr/local/bin   /usr/local/sbin   $path )
```

Solaris installs perl in /usr/bin, so it may be necessary to edit the first line in the perl scripts for the egd package to point to /usr/bin/perl, not /usr/local/bin/perl. The egd perl scripts to be modified are:

/usr/local/bin/egc.pl and /usr/local/bin/egd.pl.

Start the entropy (prngd) for use by openssl and openssh by the following steps:
Select several large text or log files to generate the initial prngd-seed, and start up the prngd daemon to start generating entropy.

```
cat   /var/adm/messages  /var/sadm/install/contents > /usr/local/etc/prngd/prngd-seed
mkdir  /var/spool/prngd
/usr/local/bin/prngd   /var/spool/prngd/pool
```

Check to see if prngd is generating entropy:

```
/usr/local/bin/egc.pl   /var/spool/prngd/pool  get
```

If it is working properly, a message similar to "<large number> bits of entropy in pool" should display.

Add the appropriate script for starting and stopping *prngd* to */etc/init.d* and create the link in */etc/rc2.d* and */etc/rc0.d*. Appendix A contains a copy of the script obtained from the OpenSSh Installation Notes -"Installing OpenSSH Packages" at http://www.sunfreeware.com/openssh.html.

After creating the prngd script in /etc/init.d, set the ownership and permissions and create the links.

```
chown  root:sys   /etc/init.d/prngd
chmod  555    /etc/init.d/prngd
ln  -s    /etc/init.d/prngd    /etc/rc2.d/S98prngd
ln   -s   /etc/init.d/prngd     /etc/rc0.d/K01prngd
```

Test stopping and starting prngd using the new script.

```
/etc/init.d/prngd   start
ps  -ef | grep prngd
```

No prngd process should be displayed.
Now start it again:

```
/etc/init.d/prngd   stop
ps  -ef | grep prngd
```

The running process should be displayed.

TCP_Wrappers

Tcp_wrappers can be used to restrict the access to specific services based on the source IP address. This can be used to limit communication through port 22, the sshd program port to specific IP (machines) on the network and thereby increase the security of the system. The system IP's are specified in the /etc/hosts.allow and /etc/hosts.deny files. If these files do not exist, access is permitted from any IP.

The tcp_wrapper package installed the daemon, 'tcpd', in /usr/local/bin. A control script to start and stop the daemon should be created in /etc/init.d, then links created to /etc/rc2.d and /etc/rc0.d for system startup and shutdown.

On this system, access to sshd will be limited to four specific machines(with static IP's) in the IT Dept. These hosts and IP's should be included in the local hosts file in full canonical format and regular format (i.e.: 129.200.10.3 buster.foo.com buster).

Create the file /etc/hosts.allow.
Edit the file and add the list of IP numbers that are allow access to sshd.
For example:

```
sshd: 129.200.2.3,129.200.2.5,129.200.2.200,129.200.2.201
```

This allows access to sshd (port 22) from the IP numbers specified.
Create the file /etc/hosts.deny.
Edit the file and enter the line:

```
sshd: ALL
```

This will deny access through sshd from any IP's that weren't specified in /etc/hosts.allow.

Installing sshd

Before starting the sshd daemon, the key information for the server must be created.

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

Create the startup/shutdown script for sshd in /etc/init.d. See Appendix B for sshd script. Change the ownership and file permissions and create the links for startup and shutdown in /etc/rc2.d and /etc/rc0.

```
chown root:sys /etc/init.d/sshd
chmod 555 /etc/init.d/sshd
ln -s /etc/init.d/sshd /etc/rc 2.d/S98sshd
ln -s /etc/init.d/sshd /etc/rc0.d/K01sshd
```

Try starting 'sshd':

```
/etc/rc 2.d/S98sshd start
ps -ef | grep sshd
```

The 'sshd' process should be listed.
Try stopping 'sshd':

```
/etc/rc0.d/K01sshd stop
ps -ef | grep sshd
```

The 'sshd' process should not be listed.

Before restarting sshd, edit the /usr/local/etc/sshd.config file and put a few restrictions on the service. We'll change or uncomment the following lines

```
Protocol 2,1
ListenAddress 0.0.0.0
LoginGraceTime 180
PermitRootLogin no
RhostsAuthentication no
RhostsRSAAuthentication no
CheckMail no
PrintMotd no
```

Restart 'sshd'.

Users

By default, Solaris installs with several user accounts that may not be needed and for security purposes should be prevented from ever getting a login shell. To secure these accounts, set the shell to /dev/null.
The user accounts to be secured are: uucp, nuucp, lp, smtp, listen, nobody4, adm, daemon, bin, nobody, noaccess.
To change the shell for each of these, run:

```
passmgmt -m -s /dev/null <user>
```

Create a username for each person who will be administering this DNS nameserver. To gain root access, they will need to use 'ssh' to login under their unprivileged 'username' and 'su ' to become superuser. The log of 'su' access will provide a good audit trail, if necessary.

Before creating the username, we will add a new unique group to /etc/group for the unprivileged users. We will create group 'itdept' with gid '51'. Lower group numbers are generally reserved for system functions.
Edit /etc/group and add a line for the user's group in the following format:

```
itdept::51:
```

For each user, select a username and uid that is distinct from any in the current /etc/password file. Create the user and home directories, then set permissions/ownership.

```
mkdir -p /home/u1/it/janedoe
useradd -d /home/u1/it/janedoe -g 51 -s /bin/sh janedoe
chown janedoe:itdept /home/u1/it/janedoe
chmod 700 /home/u1/it/jandoe
```

Create a password for IT user 'janedoe'.

```
passwd janedoe
        New password: <enter the new password>
        Re-enter new password: <again>
        passwd (SYSTEM): passwd successfully changed for janedoe
```

It will be necessary to reboot to implement the changes and test the software startup scripts that have been added and modified.
Modifications to the OBP settings can be performed by using the 'eeprom' command or at the 'ok' prompt.

```
/usr/sbin/shutdown -y –g0 –i0 "shade briefly shutting down"
```

(Shutdown system, answer 'y'es to confirm, set 'g'race period to zero seconds, set to 'i'nit state 0 stopping the operating system, broadcast the quoted message).
This will shutdown the system and bring it to an 'ok>' prompt.

OBP Settings

At the console 'ok' prompt, 'printenv' displays the OBP(Open Boot Prom) settings. Note that 'auto-boot?' is set to false, indicating that the system will not automatically boot on power-up.
Since this is a DNS server, it will need to be one of the first systems to come up after any network power outage. By setting the system to 'auto-boot', we can be assured that it will automatically attempt to boot to multi-user mode on power-up.

```
ok> setenv auto-boot? true
```

By default the 'security-mode' option is set to 'none'. This would allow any command to be typed at the console 'ok' prompt without requiring a password. There are two other

options for 'security-mode'. 'command' permits anyone to 'b'oot or 'c'ontinue at the 'ok' prompt without a password, and permits other commands but only with the OBP password. The third option is 'full', which requires a password for any command except the 'c'ontinue command.
It is important to realize that the OBP password is written to the NVRAM. If a security-mode is selected that requires a password and the password is forgotten, the system will be unusable until the NVRAM chip is replaced.
In order to permit selected IT staff access, but still provide some OBP security, the 'security-mode' will be set to 'command'. The OBP password will be stored in a secured locked cabinet, accessible to IT staff in case of emergency.

```
ok> passwd
```

The system will prompt for the password and a confirmation. This will become the OBP password.

```
ok>  setenv security-mode command
```

It is recommended that the OEM banner be replaced by a custom banner, in order to prevent system information from being displayed to potential hackers. To change the OEM banner:

```
ok> setenv  oem-banner "Authorized users only. "
```

Connect the system to the network via the ethernet interface.
Test the OBP changes:

```
ok> reset
```

After the reset the sytem should auto-boot into multi-user mode.

When the system has completely rebooted, check to see if the new processes have started:

```
ps –ef  | grep prngd
ps  -ef  | grep  tcpd
ps  -ef  |  grep  sshd
```

Now, see if we can remotely connect via ssh from one of the IP's we gave access. From one of the ssh client IP's that we allowed, login via ssh as 'janedoe'

Fix-modes

The fix-modes programs provides a way to secure our Solaris 8 operating system by checking the file permissions and ownership and correcting any security problems within the operating system files. It looks at all the files, directories and devices within the /var/sadm/install/contents and removes world and group write permissions from those files. The tar file is available from several Internet sites. We downloaded the tar from http://www.ja.net/CERT/Software/fix-modes/.
On another server with a C Compiler, create a directory and place the tar file in it.

```
 mkdir    /tools/fix-modes
 mv  fix-modes.tar    /tools/fix-modes
cd /tools/fix-modes
```

```
tar xf  fix-modes.tar
make   CC=gcc
```

This will build the binaries. Create a tar of this fix-modes directory and use 'scp' to copy it to our new server (shade.foo.com) as an unprivileged user.

```
cd ..
tar cf   fix-modes-bin.tar  fix-modes
scp  fix-modes-bin.tar  janedoe@shade.foo.com  /tmp
/bin/su
        Enter the root password.
cd  /tmp
tar xf  fix-modes-bin.tar
cd fix-modes
./fix-modes
```

All changes are listed in the /var/sadm/install/contents.mods file. Disk space should be considered before running 'fix-modes' since the output can be quite large.
If necessary, fix-modes can be 'undone' by running 'fix-modes –u'. Depending on the applications on your server, you might want to try one of the other options, such as limiting the fixes to 'Sun' packages (-S). The Readme has all the details.

Logging

In order to log successful and failed 'login' and 'su' attempts, as well as reboots and other related information, it is necessary to modify /etc/syslog.conf. By default syslogd does nothing with messages logged to LOG_AUTH.
Create the log file.

```
touch   /var/log/authlog
chmod 600  /var/log/authlog
chown   root:root /var/log/authlog
```

Edit the /etc/syslog.conf and add the following line so syslogd will use the log file.

```
auth.info        <tab>  /var/log/authlog
```

Create a '/var/adm/loginlog' file to log failed login attempts.

```
touch   /var/adm/loginlog
chmod  644 /var/adm/loginlog
chown  root:sys /var/adm/loginlog
```

Solaris Security Step By Step, 'Appendix E' provides a 'rotate' script that can be used by cron on a weekly basis to rotate logs and stop and restart the syslogd daemon. Copy the 'rotate' script and install it in /usr/local/bin/rotate. Setup a crontask to rotate logs each week.
Add the following line to the 'root' crontab file (crontab –e root) to rotate logs every Sunday at 1:30 am.

```
30 1 * * 0 /usr/local/bin/rotate  /var/log/authlog  600 4
30 1 * * 0  /usr/local/bin/rotate /var/adm/loginlog 600 4
```

The default Solaris 8 installation sets CRONLOG=YES, which will turn on logging of all crontasks.

Reviewing all of these logfiles on a regular basis can provide very useful information about several types of attacks that may have been attempted against the system, whether or not they were successful.

Patchdiag Tool

The 'patchdiag' tool "determines the patch level on your system against Sun's Recommended and Security patch list for particular hardware and OS. Additionally, it can operate from input files and will list all patches that pertain to packages installed on the system." Patchdiag can be downloaded from a secured server at http://sunsolve.sun.com and then transferred to this system, in the same manner that the patches and open source software were installed. We will run patchdiag to determine if any further patches are recommended for our system.

```
zcat /usr/local/patchdiag_1_0_4_tar.Z | tar xf –
cp patchdiag.xref /usr/local/etc
cd patchdiag-1.0.4
./patchdiag_setup
        Enter the location of the patchdiag.xref:    /usr/local/etc
```

Now run patchdiag with the '-l' option, which produces a report that lists all installed patches, uninstalled recommended patches, uninstalled security patches, uninstalled y2k patches, and other related uninstalled patches.

```
./patchdiag –l
```

If there are unistalled recommended patches on the report, determine if they should be installed by checking the readme for the patch at http://sunsolve.sun.com.
Download and install any critical patches and reboot, if necessary.

Sendmail

Solaris 8 includes sendmail 8.10.4. Although the latest version is sendmail 8.12.2, our implementation will be strictly outgoing mail(monitoring, logging) to the mailhub, so we feel sendmail8.10.4 will provide adequate security for this system.
A stripped down version of sendmail will be installed. This will use the 'nullclient' feature to create a configuration file that simply forwards all mail to a central mailhub. According to the latest Sendmail Readme at www.sendmail.org, no other features except 'nocanonify' should be used with the nullclient feature.
To create this configuration file:

```
cd  /usr/lib/mail/cf
```

Copy the subsidiary.mc file to another name – in this scenario call it 'shade.mc'
Modify 'shade.mc' to eliminate all FEATURE's except the following:

```
FEATURE(`nullclient',`mailhost.foo.com')dnl
FEATURE(`nocanonify')dnl
```

Also remove the 'MAIL' options, and be sure SMARTHOST is defined as your mailhub.
Example:

```
define(`SMART_HOST', `mailhost.foo.com')
```

where our mailhub is 'mailhost.foo.com'.

Create the sendmail.cf file from this shade.mc:

```
/usr/ccs/bin/m4 -D_CF_DIR_=/usr/lib/mail/ /usr/lib/mail/m4/cf.m4 shade.mc > shade.cf
```

Copy it into the mail directory.

```
cp ./shade.cf /etc/mail/sendmail.cf
chmod 644 /etc/mail/sendmail.cf
chown root:bin /etc/mail/sendmail.cf
```

Setup a cron task to periodically clear the mail queue, since sendmail is not running.

```
0 * * * * /usr/lib/sendmail –q
```

Now that sendmail is configured, reset the 'read-only' option in /etc/vfstab for /usr. This will prevent anyone from writing to /usr.
It should look like this:

```
/dev/dsk/c0t0d0s3  /dev/rdsk/c0t0d0s3  /usr  ufs  1  no  ro
```

Reboot to make this effective.

Other Stuff

The following security suggestions were taken from SANS Institute Track 6 Securing a Unix System Track 6.5 Linux/Solaris Practicum.
Banners and other default messages often give away information that could be useful to a potential hacker, such as an exploit in a particular version of Solaris.

1. Change the /etc/motd and /etc/issue default messages to add some type of warning for potential intruders.
2. Set the telnet BANNER to "" in /etc/default/telnetd (although we've disabled it on this system).
3. Set the ftp BANNER to "" in /etc/default/ftpd. Also set the default UMASK for uploaded files to 022 or 077.
4. In /etc/default/inetint set "TCP_STRONG_ISS = 2 " to force the system to use a better randominzing algorithg for TCP sequence numbers, making session hijacking a bit more difficult.
5. In /etc/default/login, setting the 'RETRIES' sets the number of attempts before the login program exits and the user is knocked off the system. Suggested values are 3 to 5. Also, setting SYSLOG_FAILED_LOGINS to zero will log all the failed login attempts.
6. Set the umask and path for /etc/profile and /etc/.login, the default start-up files for sh and ksh (/etc/profile) and csh (/etc/.login).

BIND

This server will be functioning as the external 'shadow' namespace for our network. By implementing a 'split horizon' or 'split namespace', only a pared down version of the organization's namespace will be visible to the Internet. This 'shadow namespace' will be located on an external 'screened' network between the Internet gateway and the internal network. A nameserver located on the internal network will contain a complete set of information on the organization hosts, but the internal nameserver will not be

accessible from the Internet. It will forward all queries, except those for hosts on its internal network, to the external nameserver in the DMZ. Only the external 'shadow namespace' in the DMZ will be accessible from the Internet. It will only contain information about hosts that are visible from the Internet. It may also contain the MX records for directory mail from the Internet to the organization's mail server, on the internal network. DNS and BIND, 4th Edition by Paul Albitz and Cricket Liu and the ISC BIND 9 Administrator Reference Manual are required reading for configuring DNS. They provide a wealth of information on securing your nameservers and on the various configurations possible.

BIND has been the target of several types of exploits over the past few years. 'Cache poisoning' and 'buffer overflows' have been the most popular types. Splitting the nameservice to limit the visibility will provide some extra protection for our network, but we also need to provide some protection for the system itself. As a first step, we will install BIND 9.2 in a 'chroot'ed jail, where it will run as an unprivileged user, instead of 'root'. By using 'chroot', nothing beyond its 'chroot'ed directory tree can be seen or accessed by BIND. This can limit any damage from an attacker, since the attacker will be captive in the chrooted jail. In addition, TSIG(Transaction signatures), available in BIND 8.2 and later, provides a means to authenticate DNS messages, including zone transfers and dynamic updates. The TSIG record becomes part of the DNS message. By using a one-way hash function,HMAC-MD5, TSIG can provide authentication and data integrity. Any change in a single bit of the input changes the hash value "dramatically and unpredictably—so unpredictably that it's 'computationally unfeasible' to reverse the function and find the input that produces the hash value." (DNS and BIND 4th Edition, "Chapter 11 Security"). The TSIG record contains a hash value that is keyed with shared secret. Verifying the hash value provides authentication of the sender, and assures the data integrity of the DNS message. The date/time portion of the TSIG record helps prevent 'replay' attacks, where a hacker captures a signed, authorized transaction and replays it later. The time is checked by the recipient to be sure that it's been received within the time allowance specified.

Chroot-BIND Installation

Since we will be running BIND in a chrooted environment, we will need to create a separate user and group for BIND, which we will call 'named'.
Create the home directory '/usr/local/dns', the group'named' and the 'named' user account.

```
groupadd  -g 200 named
mkdir   /usr/local/dns
useradd  -u 200 –s /bin/false –d  /usr/local/dns  -g 200 –c "BIND daemon" –m  named
chmod  511  /usr/local/dns
```

The shell is set to /bin/false to prevent this user from logging in. The shell must be a valid executable file. The '-m' is used to create the the home directory and add read, write, and execute permissions for the primary group.

Add ‘named’ to /etc/ftpusers to prevent ‘named’ from using ‘ftp’.

```
echo “named” >> /etc/ftpusers
```

On a secure server on your network, download the BIND 9.2.0 source and the administrator’s manual from www.isc.org. Transfer the zipped source file to /usr/local.

```
cd /usr/local
gunzip bind-9.2.0rc9.tar.gz
tar xf bind-9.2.0rc9.tar
```

The server will need the GNU ‘make’, not Sun’s /usr/ccs/bin/make. The ‘make’ package is available from from www.sunfreeware.com.

Configure and install the BIND software.

```
cd bind-9.2.0rc9
./configure
/usr/local/bin/make
```

In Sean Boran’s ‘Running the BIND9 DNS Server securely’, he recommends installing BIND into a temporary directory and then creating a tar file which we will later untar into our chrooted jail on the nameserver. Use a c-shell (/bin/csh) for the following commands and umask 027 to set default file permissions to permit group but not world access.

```
csh
umask 027
/usr/local/bin/make install DESTDIR=/tmp
cd /tmp/usr/local
# The strip command removes debugging information and symbols,
# reducing the size of the compressed file
strip bin/* sbin/* lib/*
# To reduce space further you may remove the include files
#    command: rm -rf include

# now create the compressed tar file
 tar cf - * | compress > bind9_dist.tar.Z
```

Save the tar file in a safe place and remove the temporary installation (/tmp/usr).

```
mv bind9_dist.tar.Z /opt
rm –r /tmp/usr
```

Transfer the tar file to the new system.

```
scp bind9_dist.tar.Z janedoe@shade.foo.com /tmp
```

This will do a remote SSH copy of the file to the /tmp directory of shade.foo.com as user ‘janedoe’.

Prepare the Chroot Jail

We have already created the chroot directory where we will be running BIND - /usr/local/dns.

Now create the necessary subdirectories and system files.

```
umask 022        (translates to user=rwx, group=rx, others=rx)
```

```
cd /usr/local/dns
mkdir  -p {dev,etc,lib,sbin,opt,usr,var,usr/sbin,usr/local}
chown   root:daemon {var,dev}
chmod 111 {var,dev}
mkdir  -p  var/{run,log,named}
chown named:named  var/run
chmod 700  var/run
mkdir  -p  usr/lib
mkdir  -p  usr/share/lib/zoneinfo
mkdir  -p  usr/local/{bin,lib,sbin,bind,etc}
chmod 755  usr/local/{bin,lib,sbin,bind,etc}
```

Prior to BIND 9, if BIND were being run as a unprivileged user, it would be necessary to create a password and group file in the chroot jail with entries for 'named'. BIND 9 reads the system's /etc/password and /etc/group file before calling 'chroot()', making this unnecessary.

Copy the compressed BIND tar file to the chroot jail and untar it .

```
cp /tmp/bind9_dist.tar.Z   /usr/local/dns/usr/local
cd /usr/local/dns/usr/local
zcat bind9_dist.tar.Z  | tar xvf –
```

Chroot Shared Libraries and System Files for BIND

Copy the system files needed for the chroot environment.

```
cp  /etc/{syslog.conf, netconfig, nsswitch.conf,resolv.conf, TIMEZONE}  \
/usr/local/dns/etc
```

The 'ldd' command will list dynamic dependencies of executable files or shared objects. We will use it to help determine which shared libraries we will need for the BIND chroot environment.

```
ldd   /usr/local/dns/sbin/named
```

provides the following output:

```
libnsl.so.1 =>   /usr/lib/libnsl.so.1
libsocket.so.1 =>        /usr/lib/libsocket.so.1
libpthread.so.1 =>       /usr/lib/libpthread.so.1
libthread.so.1 =>        /usr/lib/libthread.so.1
libc.so.1 =>     /usr/lib/libc.so.1
libdl.so.1 =>    /usr/lib/libdl.so.1
libmp.so.2 =>    /usr/lib/libmp.so.2
/usr/platform/SUNW,Ultra-250/lib/libc_psr.so.1
```

Note that the final library listed is sometimes dependent on the platform so this may be different depending on the hardware of your system.
Make the appropriate platform directory within our chroot, set the permissions and copy the platform library, if needed.

```
cd /usr/local/dns
mkdir –p  usr/platform/SUNW,Ultra-250/lib
chown –R root:daemon  usr/platform
chmod –R 111 usr/platform
cd usr/platform/SUNW,Ultra-250/lib
cp   /usr/platform/SUNW,Ultra-250/lib/libc_psr.so.1    .
chown  root:daemon   libc_psr.so.1
chmod 555    libc_psr.so.1
```

We can use the following commands to create a list of the needed libraries and copy these libraries with their current file attributes to our chrooted environment.

```
ldd   /usr/local/dns/sbin/named  |  awk -F\> '{print $2}'>  /tmp/ldd-out
```

Edit the 'ldd-out' to remove the platform dependent libraries we have already copied. It is recommended that we also add /usr/lib/ld.so.1 and /usr/lib/nss_files.so.1 to the list. (Boran, Sean "Running the BIND9 DNS Server securely" p4).

```
echo  " /usr/lib/ld.so.1" >>  /tmp/ldd-out
echo  " /usr/lib/nss-files.so.1" >> /tmp/ldd-out
```

Copy the listed libraries to the chrooted jail and set the permissions for read/execute only.

```
cd  /usr/local/dns/usr/lib
foreach   j  ( `cat /tmp/ldd-out`)
 ?  cp   $j   .
? end
chmod   555  *
chown   root:daemon *
cd  /usr/local/dns
rm  /tmp/ldd-out              (clean up temporary files)
```

Copy the timezone files into our chrooted jail.

```
cd   /usr/local/dns
cp –p  /usr/share/lib/zoneinfo/EST       ./usr/share/lib/zoneinfo/EST
```

Copy /usr/sbin/named-xfer to our chrooted jail. It's needed by the 'named' process for certain types of zone transfers.

```
cp   /usr/sbin/named-xfer  ./usr/sbin
```

In Rob Thomas' "Secure BIND Template Version 2.0 ", the creation of the following '/dev' files are recommended, which will be necessary for system support. The device files are 'character special files' and require a major and minor number to be specified when they are created (mknod). To find the correct major and minor numbers, change to the /dev directory and follow the links to the correct file in /dev/devices/pseudo. An 'ls –l' will display the major and minor numbers for the file. Use the same numbers when creating the file.

```
cd   /usr/local/dns/dev
```

```
mknod   conslog  c 21 0
mknod    log   c   21  5
mknod    syscon  c  0  0
mknod    zero  c  13  12
```

Zone transfers require that a null device file (/dev/null) exist.

```
mknod  /usr/local/dns/dev/null   c  13   2
```

Several sources additionally recommend creating /dev/tcp and /dev/udp devices.

```
mknod   tcp   c  42   0
mknod   udp  c  41  0
```

Set the permissions.

```
chgrp  sys  zero
chmod 620 syscon
chgrp tty  syscon
chgrp sys conslog
```

Configuring BIND

It is necessary to create a directory for the zone files used by the name server.

```
cd /usr/local/dns
mkdir –p  var/namedb
chmod –R 111  var
```

Create the configuration files for BIND 9. Abitz and Liu's DNS and BIND, 4th Edition provides all the gory details for configuring BIND 9. We will be using a fairly simple configuration.

```
cd  /usr/local/dns/etc
touch named.conf
chown  root:named  named.conf
chmod 644  named.conf
```

For purposes of this example we will use the following information:

External nameserver (this server): 129.200.9.1
Internal nameserver : 129.200.10.5
Internal networks: 129.200.10.0/24 , 129.200.11.0/24
Other hosts in DMZ: 129.200.9.13 , 129.200.9.14

There are extensive sources for configuring DNS and this document will not get into the details of the possible configuration options. Albitz and Liu'sDNS and BIND 4th Edition provides a wealth of information and is necessary reading for anyone involved in the BIND 9 configuration, as well as the ISC BIND 9 Administrator Manual available with the BIND distribution.

To get started, we will use the following /etc/named.conf for configuring an external DNS server.

```
//External  DNS server named.conf   configuration

acl internals {129.200.10.0/24; 129.200.11.0/24; };   //defining our internal networks
acl externals {129.200.9.1; 129.200.9.13; 129.200.9.14; };  // defining DMZ hosts
```

```
options {
  directory “/var/namedb”;                  // zone files directory within chroot
  pid-file "/var/run/named.pid";
  listen-on { 129.200.9.1; 127.0.0.1; };    // the external IP and the loopback address
  allow-transfer { none; };                 //  set default to not allow zone transfers
  allow-query { internals; externals; };    // restrict query access to our network
  allow-recursion { internals; externals; }; // restrict recursion to our network
  version "DNS server";                     // hide BIND version
};
zone "." {
        type hint;
        file "db.hints";                    // contain list of Internet root servers
};
zone "0.0.127.in-addr.arpa" {
        type master;                        // local loopback
        file "db.127.0.0";
        allow-query {127.0.0.1;};
};
zone "foo.com" {
  type master;
  file "db.foo.com";
  allow-query { any; };                     // allow queries from anywhere
  allow-transfer { internals; externals; }; //overrides options allow-transfer
};
 zone "200.129.in-addr.arpa" IN {
        type master;                        // reverse lookups
        file "db.129.200";
        allow-query {any; };
        allow-transfer {internals; externals;};
};
```

This configuration assumes the internal name server has been setup to forward queries for information on addresses outside the internal network to this external name server.
A sample copy of the ‘hints’ file db.hints is contained in Appendix D. This lists all the Internet root servers. Copies can be obtained from Internic.
Set the resolv.conf to search our internal nameserver(s) since it contains information about all of the hosts on our internal network. A sample is contained in Appendix E.

We will also need to create the DNS data files in var/namedb.
‘db.127.0.0’ will contain the information for our localhost loopback.
A miminal ‘db.foo.com’ will contain IP and host information for the files we want to be visible to the Internet. ‘db.192.200’ will contain the reverse lookup entries our Internet visible hosts.
The zone data files are unique for each installation. BIND 9 offers many new security features that may be applicable for your site. Sample zone data files are provided in

Appendix F. 'Running the BIND9 DNS Server Securely' by Sean Boran explains in detail the methods for reducing the risk of running BIND.
After the zone files are placed in /usr/local/dns/var/namedb, we can check the syntax by running named-checkconf.

Check the configuration, but you need to do this as 'named' or the zone data files won't be found.

```
set dns='/usr/local/dns/'
/usr/sbin/chroot  $dns  /usr/local/sbin/named-checkconf   /etc/named.conf  - u named

/usr/sbin/chroot  $dns  /usr/local/sbin/named-checkzone "0.0.127.in-addr.arpa"  \
/var/namedb/db.127.0.0     - u    named

/usr/sbin/chroot  $dns  /usr/local/sbin/named-checkzone "foo.com"  \
/var/namedb/db.foo.com  \
  -u named

/usr/sbin/chroot  $dns  /usr/local/sbin/named-checkzone " 200.129.in.addr.arpa"  \
/var/namedb/db.129.200      –u named
```

It's safe to ignore complaints regarding missing SOA and NS records for 'db.foo.com' at this point.

Sean Boran's "Running BIND9 DNS Server Securely" and Scott Wunsch' "Chroot-BIND HOWTO" provide details on securing the permissions in our chrooted BIND jail.

```
        csh
        set  dns='/usr/local/dns'
        cd  $dns                ( the top of the 'chroot'ed jail)
```

Give 'named' ownership of all the files in the jail.

```
        chown –R named  *
```

Allow named access to the configuration file – etc/named.conf.

```
        chgrp named  $dns/etc
        chown root:named  $dns/etc/named.conf
        chmod 755   $dns/etc
        chmod 640  $dns/etc/named.conf
```

Restrict access to var, opt and usr.

```
        chmod  -R g-w  var
        chmod –R  a-w  opt usr
        chmod –R  o-rwx  *  $dns/usr
```

'Root' needs to own the zone data files, but allow user 'named' to read them.

```
        chown –R root:named  $dns/var/namedb
        chmod 750   $dns/var/namedb
        chmod –R  go-w  $dns/var/namedb
```

Create log and pid files and make them accessible to the 'named' user/group for writing.

```
        touch  $dns/var/log/all.log   $dns/var/run/named.pid
```

```
chown named:named   $dns/var/log/all.log  $dns/var/run/named.pid
chmod 770  $dns/var/log/all.log   $dns/var/run/named.pid
chown  -R   named  $dns/var/log/all.log  $dns/var/run/named.pid
chown –R o-rwx $dns/var/run   $dns/var/log
```

The startup script, /etc/init.d/dns (in Appendix G) starts up the nameserver in a 'chroot'ed environment as user 'named using the command where '-u 200'is the uid of user 'named' and the '-t' indicates the chroot base direcectory.

```
/usr/local/dns/usr/local/sbin/named -u 200 -t /usr/local/dns
```

After installing the script set the permissions and the appropriate links.

```
chmod 744  /etc/init.d/dns
chown root:sys  /etc/init.d/dns
ln –s  /etc/init.d/dns   /etc/rc2.d/S72dns
ln –s /etc/init.d/dns   /etc/rc0.d/K50dns
```

Start BIND and test it out.

```
/etc/init.d/dns  start
```

To check if the process is running under user 'named':

```
ps –ef | grep named
```

produced the following output:

```
named  9369   1  4 13:46:04 ?   0:01 /usr/local/dns/usr/local/sbin/named -u 200 –t
/usr/local/dns
```

This indicates that nameserver is running under user 'named'.

Check the the logs for activity and/or errors with 'tail /var/adm/logs'.

Now try 'dig' to see what we have:

```
/usr/local/dns/usr/local/bin/dig  foo.com
```

produces the following input:

```
./dig foo.com
; <<>> DiG 9.2.0rc9 <<>> foo.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: REFUSED, id: 47072
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;foo.com.                IN      A

;; Query time: 86 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Feb  6 12:52:56 2002
;; MSG SIZE  rcvd: 30
```

At this point it's necessary to go to the various BIND reference and see what the output should be. Try other 'dig' options to test the validity of your named.conf and zones files. You will need to test whether the server will resolve Internet addresses and external network addresses for the internal and external networks, and hide information on our internal nameserver from the Internet.

A word about TSIG

To enable the TSIG features of BIND to secure transactions such as zone transfers between a master and a slave name server, it would be necessary to configure a common key on both nameservers. DNS and BIND 4th Edition has all the details. We're not going to go through those steps. Basically, you create a key on each server using the dnssec-keygen program.

```
dnssec-keygen -a HMAC-MD5 –b 128 -n HOST site1.foo.com
```

This creates the key files, for example, ' Ksite1.foo.com.+157+29995.key' and 'Ksite1.foo.com+157+29995.private'. The twe key files have a different formats, but the key can be extracted from either. A 'key' statement which includes the 'secret' is used to create the common key for a server. For example:

```
key site1.foo.com .{
        algorithm hmac-md5;
        secret "afsdjlkfadslkj;fasljkfasdjlk;sa";
};
```

where the 'secret' is extracted from one of the key files that were generated.
A 'server' statement with a 'keys' substatement indicates that the nameserver will sign all requests to a particular remote server. The 'zone' statement can restrict zone transfers to those signed with a specified TSIG key.
There have been a few problems reported when nameservers did not have their time synchronized, but those are easily corrected if NTP is implemented.

Since it is only available on BIND8.2 and later, if some of your nameservers are not up to that level, TSIG would not be an option for transactions with those servers. It does provide a much higher level of security for any name server transactions when implemented.

MAINTENANCE

Backup

Backups are a critical piece of the security policy for any system. They provide a reference point, should any serious problems occur. A good backup will need to be taken with the system in as quiet a state as possible. For this we will boot to single-use mode, then 'fsck' the systems and mount all the file systems.

```
reboot -- -s
fsck
```

mount –a

Be sure the tape is rewound.

mt –f /dev/rmt/0 rewind

We'll do a level 0(all files) 'ufsdump' to backup all the file systems to tape, using the 'norewind' option for the tape drive.

```
for filesys in / /usr /var /opt /usr/local
    /usr/sbin/ufsdump 0f /dev/rmt/0n $filesys
done
```

Now rewind and unload the tape.

mt –f /dev/rmt/0 rewoffl

In Solaris Security Step by Step, it is recommended that the backup be repeated on new media as an extra precaution.
When it completes, reboot the system to multi-user mode 'reboot'.

Create a script similar to the code above to dump all the file systems to tape. Additional code to keep logs of the backups should be added. An example is in Appendix H. Add this script to the root crontab and schedule it to run each night. Tapes will be removed to a secure location on a daily basis. Use 'crontab –e root' to edit the crontab file and add the following entry for our 'backup' script.

10 1 * * * /bin/csh –c /usr/local/sys/dump.sys.list

lsof

'Lsof' is a tool that is used to see which files are opened by which processes. It's available as a package from www.sunfreeware.com and provides a critical piece for auditing our systems, tracking down a hacker process, and various other uses.
It's a 'must have' tool.
Download the lsof-4.49-sol8-sparc-64-local.gz into /opt.

gunzip lsof-4.49-sol8-sparc-64-local.gz

pkgadd –d lsof-4.49-sol8-sparc-64-local

This will install the lsof binary into /usr/local/bin.

lsof –p <pid>

will show all the files open for that process id.

lsof –u janedoe

will show all the files open by user janedoe.

lsof +L1

displays files that are open but have become unlinked from the file system A process deleted the file, but the program still has it open. Normally, this should return nothing. If you get a result, you can kill the process(es) and reclaim the disk space. Hackers use this trick to delete their log files and other data.
Try it out and become familiar with the various options. They can be very useful.

Watcher

In order to pick out the important events in large log files, a tool called Watcher was developed by Kenneth Ingraham. According to the Computer Incident Advisory Capability website, http://ciac.llnl.gov/ciac/ToolsUnixSysMon.html#Watcher, Watcher is a "system monitoring tool that issues a number of user-specified commands, parses the output, checks for items of significance, and reports them to the system administrator." Through a set of rules you can specify which lines in the log are 'interesting' and those that are 'uninteresting'(throw away). The download is freely available from the CIAC website. We plan to install this as soon as possible.

Tripwire

Tripwire is a file system integrity-checking program. It requires that we build a configuration file that will designate which files and directories we want to check, and the attributes that we want to verify. It initially builds a database of checksums that are used to compare against for the subsequent runs. It is recommended that the configuration files and database be kept on some type of removeable media, such as a floppy disk, writeable CD-ROM or tape that can be write-protected to secure it against any type of corruption.

To install tripwire you need the latest source (Tripwire-1_3_1-1_tar.gz), available from http:/www/tripwiresecurity.com/products/index.cfml. You'll also need the following packages installed on the server where you will build Tripwire:

MD5 – a cryptographic checksum program
Gzip – to uncompress the downloaded file
PGP - to verify the authenticity of the software distribution
C-Compiler - either GNU gcc or the Sun C Compiler.

The CERT Coordination Center provides excellent document on implementing Tripwire at http://www.cert.org/security-improvement/implementations/i002.02.html.

Choose a directory on the development server with sufficient free disk space.

After you have unzipped and untarred the distribution, copy the default Solaris 2.x Tripwire configuration from the ./config directory to /etc.

Edit the ./include/config.h file and adjust it to for your system. You will need to specify the device (RW CD-ROM, floppy) that you will be using to store the Tripwire databases.

Create an initial configuration file.

```
cp ./config/tw.conf.sunos5 /etc/tw.config
```

Edit the tw.config file to include any additional local binaries or other files that you want to monitor.

Now build the Tripwire executable by running 'make'.

Create a tar file of the Tripwire directory and move it to the secure system.

Install the Tripwire distribution by running 'make install' or manually copy the following Tripwire files to the appropriate places.

```
cp man/siggen.8 /usr/local/man/man8/
cp man/tripwire.8 /usr/local/man/man8/
cp man/tw.config.5 /usr/local/man/man5/
cp src/tripwire /usr/local/bin/
```

cp src/siggen /usr/local/bin/

To check the build, run ' make test'.

There are four modes of running Tripwire:

1. database generation
2. database update
3. interactive update mode
4. integrity checking

We have a portable SCSI floppy drive that we will be using for Tripwire.

mount –n /dev/disk /floppy

/usr/local/bin/tripwire -initialize

This creates the initial database that will be used later by Tripwire for file-integrity comparisons. We will also add the configuration file and the tripwire binary to the floppy.

cp /etc/tw.config /floppy

cp /usr/local/bin/tripwire /floppy

umount /floppy

Now write-protect the disk to disable writing to it. Store this disk in a secure location. Make an exact copy of this floppy on new media to be used as our working copy. Be sure to also write-protect this copy.

Tripwire file-integrity checking should be done on a regular schedule.

Mount the working copy of the database and run the tripwire binary from the floppy.

mount –n /dev/disk /floppy

cd /floppy

./tripwire -c ./tw.config -d ./database/tw.db_<the local hostname>

umount /floppy

It will report any unexpected changes in the files that you have specified to be checked. Thoroughly investigate and if appropriate, initiate your incident response procedures if any unexplained changes have occurred.

Secure the floppy.

Verbose mode lists all the files as Tripwire scans them. 'Phase 4' summarizes the findings. It lists Total files scanned, files added, files deleted, files changed and changes discarded after applying the 'rules' from the configuration.

Checking our Configuration

We've been checking as we went along, but it's a good idea to test again, and again. We'll follow the checklist that was used in SANS Institute's 6.5 Linux/Solaris Practicum. and added a few of our own.

1. Can we login using telnet or rsh? From a remote host, try to 'telnet shade.foo.com'. It should come back with "Trying ...telnet: Unable to connect to remote host: Connection refused" or something similar. This is good. It means telnet isn't working.

   Now try 'rsh shade.foo.com' from the remote host. It just hangs and eventually comes back with "shade.foo.com: Connection refused" This is also a good sign.

If we were able to get in, we would want to check and see why inetd has been started.

3. Can we use ssh from outside the specified hosts in our hosts.allow? You need to go to a machine that's not included in the hosts allow and try to "ssh janedoe@shade.foo.com". We're denied access.
   If we were able to get in, we would need to look at the hosts.allow and hosts.deny to see where the problem might be.
4. Can we use ssh to get a root login from a remote host?
   ssh root@shade.foo.com
   Yes, we can, but we shouldn't. We need to check on the "PermitRootLogins' in the etc/sshd_config. It should be set to 'no' or 'forced-commands-only' for public key authentication.
5. Can I write to /usr? Try to 'touch /usr/sadm/test'. If you can, you haven't set the read-only option in /etc/vfstab for /usr.
6. Can I run a set-uid process from /var/tmp?
   cp /usr/bin/ps /var/tmp
   cd /var/tmp
   chmod 4111 ps
   ^D
   $ /usr/bin/ps -ef
   $ /var/tmp/ps -ef gives different output indicates it's running set-uid.
   This can be corrected in the /etc/vfstab , by setting the nosuid option for /var.
7. Run 'ps –elf' and become familiar with the processes. Does anything look amiss?
8. Check the logs. Are there entries from 'sshd', 'named' and 'su'?
   We may have forgotten to add the entry to /etc/syslog.conf for auth.info.
   Other possible reasons would be misconfiguration of TCP wrappers, or ssh_config. Check them out.
9. Run 'netstat –an' to see what's connecting from where, using which ports.

Ongoing Security

Post-installation security involves continual monitoring of the system to detect any changes. Tripwire provides one means of monitoring file-integrity. Monitoring logs on a daily basis (using Watcher) is another way of assuring the integrity of our system. Nessus is network scanning tool that can be used to find the vulnerabilities in the network. It will be important to regularly audit to know where our problems lie. All of these tools can be helpful. Familiarity with the general system commands such as 'netstat', 'ps ', 'iostat', 'vmstat', 'sar', 'uptime', 'df' and all their options are extremely useful in providing information about the status of a system to the system administrator.

Conclusion

Are we done? Not a chance. New vulnerabilities seem to be part of this business. Keeping up with patches, and software upgrades and continually monitoring will be extremely important in keeping the systems secure against the latest threats, but a secure installation will always be an important first step.

Appendix A

/etc/init.d/prngd script:

```
#!/bin/sh
# prngd
#
# your name and the date installed
#
pid=`/usr/bin/ps -e | /usr/bin/grep prngd | /usr/bin/sed -e 's/^  *//' -e 's/ .*//'`
case $1 in
'start')
        /usr/local/bin/prngd /var/spool/prngd/pool
        ;;
'stop')
        if [ "${pid}" != "" ]
        then
                /usr/bin/kill ${pid}
        fi
        ;;
*)
        echo "usage: /etc/init.d/prngd {start|stop}"
        ;;
esac
```

Appendix B

/etc/init.d/sshd script:

```
#!/bin/sh
# sshd
#
# your name and the date installed
#
pid=`/usr/bin/ps -e | /usr/bin/grep sshd | /usr/bin/sed -e 's/^  *//' -e 's/ .*//'`
case $1 in
'start')
        /usr/local/sbin/sshd
        ;;
'stop')
        if [ "${pid}" != "" ]
        then
                /usr/bin/kill ${pid}
        fi
        ;;
*)
        echo "usage: /etc/init.d/sshd {start|stop}"
        ;;
esac
```

Appendix C

Log Rotation Script from SANS Institute Solaris Security Step By Step, "Appendix E"

```
#!/bin/ksh

# rotate – A script to roll over log files
# Usage:  rotate   /path/to/log/file [mode  [#revs]  ]

FILE=$1
MODE=${2: -644}
DEPTH=${3: -4}

DIR=`dirname $FILE`
LOG=`basename $FILE`
DEPTH=$(($DEPTH – 1))

if  [ ! -d  $DIR ] ; then
        echo "$DIR:  Path does not exist"
        exit 255
fi

cd   $DIR

#rotate the old logs to next depth
while  [  $DEPTH   -gt 0 ]
do
        OLD=$( ($DEPTH –1))
        if [ -f $LOG.$OLD ]; then
                mv   $LOG.$OLD   $LOG.$DEPTH
        fi
        DEPTH=$OLD
done

# rotate current log to log.0
if  [  $DEPTH –eq 0 –a –f   $LOG ]; then
        mv   $LOG   $LOG.0
fi

# create the new log and set the permissions
cp  /dev/null   $LOG
chmod $MODE   $LOG

# stop and start syslogd  so it that will use the new log
/etc/rc2.d/S74syslog stop
/etc/rc2.d/S74syslog start
```

Appendix D

Sample root hints file 'db.hints'

```
.                       6D  IN      NS      A.ROOT-SERVERS.NET.
.                       6D  IN      NS      B.ROOT-SERVERS.NET.
.                       6D  IN      NS      C.ROOT-SERVERS.NET.
.                       6D  IN      NS      D.ROOT-SERVERS.NET.
.                       6D  IN      NS      E.ROOT-SERVERS.NET.
.                       6D  IN      NS      F.ROOT-SERVERS.NET.
.                       6D  IN      NS      G.ROOT-SERVERS.NET.
.                       6D  IN      NS      H.ROOT-SERVERS.NET.
.                       6D  IN      NS      I.ROOT-SERVERS.NET.
.                       6D  IN      NS      J.ROOT-SERVERS.NET.
.                       6D  IN      NS      K.ROOT-SERVERS.NET.
.                       6D  IN      NS      L.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     6D  IN      A       198.41.0.4
B.ROOT-SERVERS.NET.     6D  IN      A       128.9.0.107
C.ROOT-SERVERS.NET.     6D  IN      A       192.33.4.12
D.ROOT-SERVERS.NET.     6D  IN      A       128.8.10.90
E.ROOT-SERVERS.NET.     6D  IN      A       192.203.230.10
F.ROOT-SERVERS.NET.     6D  IN      A       192.5.5.241
G.ROOT-SERVERS.NET.     6D  IN      A       192.112.36.4
H.ROOT-SERVERS.NET.     6D  IN      A       128.63.2.53
I.ROOT-SERVERS.NET.     6D  IN      A       192.36.148.17
J.ROOT-SERVERS.NET.     6D  IN      A       198.41.0.10
K.ROOT-SERVERS.NET.     6D  IN      A       193.0.14.129
L.ROOT-SERVERS.NET.     6D  IN      A       198.32.64.12
```

Appendix E

Sample /etc/resolv.conf:

```
domain foo.com
nameserver 129.200.10.5
nameserver 12.127.166.166
nameserver 12.127.177.177
```

Appendix F
Sample zone files

1.Sample 'db.127.0.0' reverse loopback file:

```
$TTL 3D
@           IN      SOA     shade.foo.com. root.shade.foo.com. (
                            1       ; Serial
                            8H      ; Refresh
                            2H      ; Retry
                            4W      ; Expire
                            1D)     ; Minimum TTL
                    NS      shade.foo.com.
1                   PTR     localhost.
```

2. Sample 'db.foo.com' zone data file.

```
$TTL    1D
foo.com.        IN  SOA  shade.foo.com root.shade.foo.com   (
                            200202041           ; serial
                            1H                  ; refresh
                            1H                  ; retry
                            1W                  ; expiry
                            1D )                ; minimum
;
                    IN      NS      shade.foo.com.
                    IN      NS      ns1.someisp.net.      ;    ISP nameserver
                    IN      MX      10  mailhost.foo.com.    ; mail hub
                    IN      MX      20  mailhost2.foo.com.  ;  secondary mail hub
;
$ORIGIN  foo.com.
localhost       IN      A       127.0.0.1
shade           IN      A       129.200.9.1
mailhost        IN      A       129.200.10.4
mailhost2       IN      A       129.200.11.4
*               IN      MX   10 mailhost.
```

3. Sample 'db.129.200' file (reverse lookups).

```
$TTL    1D
200.129.in-addr.arpa.   1D IN SOA  shade.foo.com root.shade.foo.com   (
```

```
                        200202041           ; serial
                        3H                  ; refresh
                        1H                  ; retry
                        1W                  ; expiry
                        1D )                ; minimum

200.129.in-addr.arpa.       NS      shade.foo.com.
200.129.in-addr.arpa.       NS      ns1.someisp.net.    ;   ISP slave nameserver

1.9.200.129.in-addr.arpa.   PTR     shade.foo.com.      ;  external nameserver
4.11.200.129.in-addr.arpa   PTR     mail.foo.com.       ; mail hub
6.9.200.129.in-addr.arpa.   PTR     www.foo.com.        ; external web server

; enter other local network computers that need reverse lookup
```

Appendix G

/etc/init.d/dns startup script

```
#!/bin/sh
#
# starting named
#  bind
chroot="/usr/local/dns";
named="/usr/local/sbin/named";
pid="$chroot/var/run/named.pid";
case "$1" in
'start')
        [ -f $pid ] && kill `cat $pid` >/dev/null 2>&1
        if [ -f $chroot/usr/local/sbin/named -a -f $chroot/etc/named.conf ]; then
           echo 'starting internet domain name server.'
          (umask 027;  $chroot/usr/local/sbin/named  –u 200  –t $chroot)
        fi

;;

'stop')
        echo 'Stopping name server'
        kill `cat $pid`
        if [ "$?" -ne 0 ]; then
          echo "Warning: name server not stopped"
        else
          echo "nameserver stopped."
        fi
;;
*)
        echo "Usage: $0 { start | stop }"
        ;;
esac
```

Appendix H

Sample backup script:

```
#!/bin/csh
# backup script -  dumps and logs file systems from list
# called  sys.fs.list'
#
set host=`/bin/hostname`
set logname=/var/logs/dumplogs/`date '+%y%m%d'`.$host.log
echo "Dump.sys.list executing on `date`" > $logname
set nofiles=`cat /usr/local/sys/backup/sys.fs.list | wc -l`

# rewind tape before we start
mt -f /dev/rmt/0c rewind
# dump each file system from sys.fs.list and send standard output and
# standard error to the dump.log file
foreach i (`cat /usr/local/sys/backup/sys.fs.list`)
      echo "Starting dump of $i at `date`"
      echo "/usr/sbin/ufsdump 0uf  /dev/rmt/0cn $i" >> $logname
      /usr/sbin/ufsdump 0uf /dev/rmt/0cn $i >>& $logname
      echo "Finished  dump of $i at `date`" | tee -a $logname
end
```

References

Acheson, Steve; Green, John; Pomeranz, Hal. SANS Institute Track 6Securing Unix Systems 6.3 Topics in Unix Security. October 2001.

Albitz,Paul and Liu,Cricket. DNS and BIND, 4TH Edition, May 2001. "Chapter 11 Security". http://www.oreilly.com/catalog/dns4/chapter/ch11.html

Albitz,Paul and Liu,Cricket. DNS and BIND, 4TH Edition. April 2001. O'Reilly.

Allman, Eric "Sendmail Features" http://www.sendmail.org/m4/features.html

Boran, Sean, "Hardening Solaris". September 5, 2001. http://www.boran.com/security/sp/Solaris_hardening2.html

Boran, Sean. "Running the BIND9 DNS Server securely" Nov. 6th 2001. http://www.boran.com/security/sp/bind9_20010430.html#introduction

Brotzman,Steve; Pomeranz,Hal. SANS Institute Track 6Securing Unix Systems 6.5 Linux/Solaris Practicum October 2001.

Brotzman,Steve; Pomeranz,Hal. SANS Institute Track 6Securing Unix Systems 6.4 Running Unix Applications Securely. October 2001.

Chouanard, Jean. "How to install Solaris and have a good host security." November 19 2000 VERSION= yassp v0 beta#15, Release candidate#2 http://www.yassp.org

Christensen, Steve. "Installing OpenSSH Packages". © Copyright 2001 Steven M. Christensen and Associates, Inc. Updated December 22, 2001. http://www.sunfreeware.com/openssh.html

Herrmann, Michael. "Running BIND chroot on Solaris" 2000/06/21 http://home.in.tum.de/~herrmanm/solaris/bind.html

Ingraham, Kenneth. "Watcher". http://ciac.llnl.gov/ciac/ToolsUnixSysMon.html#Watcher

Langfeldt, Nicolai and Norrish, Jamie and others. DNS HOWTO. http://www.linuxdoc.org/HOWTO/DNS-HOWTO.html#toc6

Liu, Cricket. "Upgrading to BIND 9: The Top Nine Gotchas". http://sysadmin.oreilly.com/news/dnsandbind_0401.html

Liu, Cricket. "Men & Mice "Cricket's DNS Corner" http://www.menandmice.com/9000/9310_DNS_Corner_Q&A.html#Q4

References (cont'd)

Lugo, Dave. "Dual chrooted BIND/DNS servers" Version 1.03 updated March 5, 2000. http://www.etherboy.com/dns/chrootdns.html

Pomeranz, Hal. "Enhancing DNS Security: BIND and chroot()". February 27, 2001. http://www.8wire.com/articles/print_article.asp?printAID=1752

Pomeranz, Hal. SANS Institute Track 6 Securing Unix Systems 6.1 Common Issues and Vulnerabilities in Unix Security October 2001.

Pomeranz, Hal. SANS Institute Solaris Security Step By Step.v. 2.0 2001.

Pomeranz, Hal. SANS Institute Track 6 Securing Unix Systems 6.2 Unix Security Tools October 2001.

Quinton, Reg http://www.samag.com/print/documentID=15712

Reid, Jason; Watson,Keith. "Building and Deploying OpenSSH for Solaris Operating Environment" Sun BluePrints OnLine July 2001. http://www.sun.com/blueprints/0701/openSSH.pdf

Seng Chor, Lim. http://rr.sans.org/DNS/alternatives.php - "DNS Security Considerations and the Alternatives to BIND", October 2, 2001.

Shostack, Adam, "Chrooting DNS server on Solaris". http://www.kanga.nu/~mirror/mirrors/www.homeport.org/%257Eadam/dns.html

Strong, Mark "Defense In-Depth on a Solaris 2.X System: A Resource Guide. July 1, 2001. http://rr.sans.org/unix/solaris_2x.php

Thomas, Rob. "Secure BIND Template Version 2.0 chroot Jail for BIND on Solaris 7 Version 1.1." 20 DEC 2000. http://www.cymru.com/~robt/Docs/Articles/secure-bind-template.html

Thomas, Rob. "UNIX IP Stack Tuning Guide v2.7". 03 DEC 2000. http://www.cymru.com/~robt/Docs/Articles/ip-stack-tuning.html

Wiseman, David, http://www.linuxdoc.org/HOWTO/DNS-HOWTO.html#toc6

Wunsch,Scott. "Chroot-BIND HOWTO" v1.5 December 1, 2001. http://www.linux.org/HOWTO/Chroot-BIND-HOWTO.html

"Ask Mr. DNS" Copyright © 1997-2000 Acme Byte & Wire LLC. http://www.acmebw.com/askmrdns/

References (cont'd).

BIND 8 Security FAQ http://www.nominum.com/resources/alerts/security-faq.html

BIND 9 Administrator Reference Manual Internet Software Consortium. Copyright 2001. http://www.nominum.com/resources/documentation/Bv9ARM.pdf

"CERT® Advisory CA-2001-02, Multiple Vulnerabilities in BIND. Original release date: January 29, 2001. Last revised: August 07, 2001. Source: CERT/CC" http://www.cert.org/advisories/CA-2001-02.html

"CERT® Security Improvement Modules" http://www.cert.org/security-improvement/#unix

"Detecting Signs of Intrusion" Copyright 2000 Carnegie Mellon University. CERT Coordination Center. http://www.cert.org/security-improvement/modules/m09.html

"Develop a computer deployment plan that includes security issues."
CERT Coordination Center.
http://www.cert.org/security-improvement/practices/p065.html

"Domain FAQTS". http://www.faqts.com/knowledge_base/view.phtml/aid/8895/fid/699

"DNS Spoofing". Men & Mice
http://www.menandmice.com/9000/9211_dns_spoofing.html

"Installing, configuring, and using Tripwire® to verify the integrity of directories and files on systems running Solaris 2.x" CERT Coordination Center. Copyright 2000 Carnegie Mellon University. http://www.cert.org/security-improvement/implementations/i002.02.html

"L-030: Four Vulnerabilities in ISC BIND". January 29, 2001 18:00 GMT.
CIAC, U.S. Dept. of Energy, Computer Incident Advisory Capability, Information Bulletin. http://www.ciac.org/ciac/bulletins/l-030.shtml

"Securing Network Servers" CERT Coordination Center
http://www.cert.org/security-improvement/modules/m10.html#s.improvement

"Diagnostic Tools, Patchdiag Tool". Sunsolve Online. Sun Microsystems.
http://sunsolve.Sun.COM/private-cgi/show.pl?target=resources/patchdiag