Global Information Assurance Certification Paper

Implementing a Inexpensive Small business Internet services server on Solaris 9 Sparc

Mark Hammerschmith
GCUX Practical Assignment Version 1.9, Option 1
Submitted 14 August 2003
Securing UNIX Step By Step

Introduction & System Description:

The budget for the hardware was limited, and Sun Sparc hardware was required. To keep costs minimal, a pair of Sun Fire V100 systems were chosen to host the applications. The systems are the base model, with single 500 Mhz UltraSPARC-IIi processor, 512 Meg ram, and a 40GB IDE drive.

The OS chosen was Solaris 9. The software required to host the services was Apache, BIND, Sendmail. To connect to the system to update web content OpenSSH will be installed. SUDO was installed as an administration tool, and Perl was determined to be a necessary tool for web developers. NTP was also considered necessary to provide accurate time. Aide and Tripwire were desired to monitor for changed files, but requires Gmake and GCC, and we are limiting the installed applications to sun C compiled software. Installing and supporting GCC in addition to CC

The goal is to build a "public system" hosting all of the applications, and secure it as much as possible. Having the services separated onto multiple servers may have made each one easier to secure. However, the low bandwidth of the internet connection means the traffic to the system should not be enough to overload it. And securing a single system rather than multiple makes fewer points of compromise available.

The second server, referred to as the "private systsem" will be used to build and test applications/updates before moving them to the public server.

The public system will have a routable IP address and reside in a DMZ. The private system does not have a routable address, and will remain behind the firewall on an internal network. The build of the private system is not covered here, other than in reference to where it is used to compile software.

The layout chosen to secure the public system was to install only the necessary software. The private system was installed with all the tools necessary to build, compile and test software. Once the software was ready, it is moved to the public system.

Risk concerns of System:

The primary risk of the server was the limited time provided to get the system up and running. Some options, such as chroot of Apache and Bind, were investigated but thrown out as not providing enough security enhancements to warrant the time investment at the time.

The large number of services and broad access cause potential problems in the services available a concern. Many of the valid users who will have access to the system are non-technical, so preventing accidental damage is also a concern.

That said, risk analysis of the system was categorized in four areas.

Physical: The hardware itself is susceptible to a variety of potential problems.

1) Physical Security:
   Is the system itself secure. If someone walks off with the system, or pours a cup of coke on it any effort to secure the software will not help.
2) Power Problems:
   Loss of power to the system not only will mean it is down while the power is out, but can also cause corrupt files and/or hardware damage.
3) Network Connectivity:
   If the network connection is damaged or removed the system obviously wont be up. Further, someone could then take over the IP address and intercept mail, provide a defaced website, etc.

External attacks: These we identified as attacks by those who do not have an account to access the system. Each network service on the system had to be evaluated.

1) Can the service be remotely compromised to gain access
   Services listening on the network will give some level of access to the system. Software mis-configuration or bugs, such as buffer overruns, could be used remotely to gain access to the system.
2) Is the service susceptible to DOS:
   Any network service will be susceptible to a DOS attack. Configuration or software bugs that make it further susceptible can be prevented.
3) Can the service be remotely exploited (used for DDOS, Spam relay, etc).
   Network services need to be configured and patched to prevent illegitimate use where possible. It needs to be verified that Sendmail can't be used as an open relay, BIND configured to prevent external recursive lookups,

Internal compromises: These we identified as coming from someone who has, or gains access to, an account on the system. These accounts should be restricted not just in protection against attacks, but also against accidental damage.

1) Can they escalate permissions:
   Can someone who has access to the system use tools to escalate their permissions.
2) What can access do they have:
   At this point the administrators are the only ones who need access to the entire system. The users will only need to be able to update content on the web site. Accomplishing this will prevent many potential problems on the system.

Policy Risks: What policy's and settings need to be put in place to make the system more secure.

1) Sharing and posting of passwords:

   Since the system will use repeated passwords, policy needs to be that users do not share or post passwords.

2) Easy passwords:

   Users tend to use passwords that are far to simple to guess. Forcing them to use longer and more complex passwords will make it more difficult for a hacker to guess them.

3) Lack of changing passwords:

   The longer a user is allowed to use the same password, the more likely that password is to be compromised. Forcing periodic change of passwords will reduce that.

4) Unsecured passwords:

   Telnet and FTP, which are commonly used to connect to servers to manage and update content, are unencrypted and easy to compromise. Any traffic that includes sending passwords over the network needs to be encrypted.

System installation Step-By-Step:

Physical installation:

The physical location of the system was to install it in a secure office in a lockable cabinet. Policy is to keep the cabinet and office locked except when in use.

A UPS was installed in the cabinet to provide continuous power in case of brownouts or blackouts.

The cabling between the system and the firewall runs through conduit, limiting potential interception of the network connection. Further, the switch is in another secure area, and was configured to only allow traffic from the public and private systems MAC address. Granted, the MAC address could be spoofed, but it's one more level of protection

The firewall and switch are managed by an external organization. All traffic to public.myorg.org is routed to the public server. All of the other systems in the office, including the private system, are NAT behind a single address. Incoming SMTP traffic is statically mapped on the firewall to route to the internal mail system, and configured to only allow connections from the public system. Other new incoming traffic is blocked from the private network.

The following installation and setup process was done with the public and private systems temporarily on a isolated network, along with the workstation used as the console.

Step by Step install Guide:

The installation is broken down into two stages, the OS install, and the application installs.

Stage 1: OS Installation:

For the initial install a serial connection to a workstation was used. The server came with Solaris 8 pre-installed, and we want to use Solaris 9. On the public server use the Solaris 9 "minimal install" option, since our goal is to have only what we need installed on that system. We have already installed our private server with a full install of Solaris and C compiler, where we will compile applications before moving them to this system.

For the install we have set up a non-public hub and connect the Windows 2000 workstation we are using for the install and the two V100's. The software and patches we need have been downloaded to the workstation.

The V100 does not come with a video console, so first it needs to be connected to a workstation using the serial port using the provided serial cable. Once connected, Launch Terminal and connect to the V100 using the settings: Bits Per Second: 9600, Data Bits: 8, Parity: None, Stop Bits: 1, Flow Control: Xon/Xoff.

Power up the V100, while the system is booting, execute a Ctrl-Break to get to the OpenBoot prompt.

Step One: Boot the CD, determine the language for the install, set up a temporary boot on the swap partition, and prepare for the actual OS install.

Insert the Solaris 9 Install CD and enter:

boot cdrom

First option is what language to use for the install, we chose to use (1) English:

The Solaris Installer can be run in English, or any of the following languages:

1) English *6) Japanese*
2) German *7) Korean*
3) Spanish *8) Swedish*
4) French *9) Simplified_Chinese*
5) Italian *10) Traditional_Chinese*
Select the language you want to use to run the installer: 1

We are then prompted with options for partitioning the drive. We want to do our own file system layout. This system will initially have several services, and potentially more in the future. Because of this and available drive space we gave the swap file 2 Gig of space. The default of putting the swap at the beginning of the drive for more flexibility is accepted, and we are ready to start copying files for the install and reboot.

Scanning system disk information...
Searching disks for upgradable Solaris root devices...
The following root devices were found to be upgradable:

Release	*Root Device*
Solaris 8	*c0t2d0s3*

Searching for locations to accommodate a temporary copy of the Solaris installation software. Swap slices are usually erased at reboot, so it is preferable to place the Solaris installation software on slice labeled swap.

No swap slices that begin at the first usable cylinder have enough space to accommodate a temporary copy of the Solaris installation software.

Using a slice that begins at the first usable cylinder allows the most flexibility during filesystem layout. If you are doing an initial install and you are not preserving any filesystems, you can re-partition a disk with the swap slice starting at the first usable cylinder.

Would you like to re-partition a disk? [y,n,?,q] y

The default root disk is /dev/dsk/c0t2d0.
The selected disk will be re-partitioned before the Solaris installation software is copied to the disk.

WARNING: ALL INFORMATION ON THE DISK WILL BE ERASED!

Do you want to re-partition /dev/dsk/c0t2d0 [y,n,?,q] y

The selected disk, c0t2d0, contains a Solaris operating environment.
If selected, the Solaris operating environment on c0t2d0 will be erased.
Continue and use c0t2d0 to install the Solaris installation software [y,n,?] y

NOTE: The swap size cannot be changed during file system layout.

Enter a swap slice size between 382MB and 38161MB, default = 512MB [?] 2048

Placing the swap slice at the beginning of the disk will allow the most flexible file system partitioning later in the installation.

Can the swap slice start at the beginning of the disk [y,n,?,q] y
Confirm Information:

Disk Slice : /dev/dsk/c0t2d0s1
Size : 2048 MB
Start Cyl. : 0

WARNING: ALL INFORMATION ON THE DISK WILL BE ERASED!

Is this OK [y,n,?,q] y

Disk slice c0t2d0s1 will be used to accommodate a temporary copy of the Solaris
installation software. After the files are copied, the system will reboot and the installation will continue. Please Wait...

Copying mini-root to local disk....done.

Copying platform specific files....done.

Preparing to reboot and continue installation.
Rebooting to continue the installation.
syncing file systems... done
rebooting...

The system is now reboots to the temporary boot installed on the Swap partition

Step Two: Install the OS, choose language, basic network information, optional packages, and drive layout.

The system reboots to the install on the swap partition. For simplicity we chose to set up networking at this point. Since we are on a private hub we are not worried about the system being compromised before we have had a chance to secure it.

First we configured the primary network interface with a static address, set the default route. We did not enable Kerberos or Name service. Name service will be enabled later, after BIND is installed on this system. The configuration summary after entering the install choices is:

You have entered the following values:

Host Name: private
IP Address: 192.168.0.19
System part of a subnet: Yes
Netmask: 255.255.255.64
Primary Network Interface: dmfe0
Enable IPv6: No
Default Route: Yes
Router IP Address: 102.1680.1
Name Service: None
Enable Kerberos: No
Time Zone: US/Pacific

We then select the options to reboot, install from CD/DVD. We then choose to do a Custom install, Set our region to North America, and select no additional

products. When it gets to the Software selection we want to choose the "Core Group".

Available Solaris Software groups:

1. Entire Group Plus OEM (1607.8 MB)
2. Entire Group (1566.7 MB)
3. Developer Group (1299.8 MB)
4. End User Group (898.0 MB)
5. Core Group (258.1 MB)

Select the number corresponding to the desired Solaris software group [2]: 5

The Core Solaris Software Group provides the minimum code required to boot and
run a networked Solaris system.

Please indicate if you want the Default Packages for the Core Group or if you
want to select Custom Packages. Selecting Custom Packages allows you to add or
remove packages from the selected Solaris Software Group. When selecting which
packages to add or remove, you will need to know about software dependencies
and how Solaris software is packaged.

1. Default Packages
2. Custom Packages

Default Packages or Custom Packages [1] 2

The Solaris Installer is gathering cluster and package information, please wait...

Select the clusters and packages you would like to install.

At this point the available packages are listed. After reviewing the 366 packages listed we decided to add NTP and remove Sendmail support. In this release NTP was package 170, so we select to toggle it on (SUNWCntp: Added package: 170, Network Time Protocol). We are going to install sendmail manually, so toggled package 231 off (SUNWCsndm: Removed package: 231, Sendmail support).

After making the custom package modification, we are now ready to lay out the disk partitions. The file system layout that was chosen was:

/ 4096MB

swap	2048MB
/usr	8192MB
/var	4096MB
/export	16384MB
/home	2048MB

/var was given 4GB to allow ample space for Log files and the Mail queue.
/export will be mounted NOSUID, and used as a CHROOT jail for non-administration users, the web site data.
/usr/local/etc was given a separate mount point so it could be mounted NOSUID, and / could be mounted as RO.

Once the file system was laid out, the summary of our choices was presented:

File System operations:

1. Print the current partition table
2. Modify a disk's partition table
3. Return to beginning
4. Done

Select the number corresponding to a file system operation, 'Return to beginning' to change selections, or 'Done' to proceed with the install [4]: 4

The following items will be installed:

Solaris Operating Environment: Solaris 9 Software 12/02
Solaris Software Group: Core Group
Root Device: c0t2d0
File Systems:
c0t2d0s0 / 4096 MB
c0t2d0s1 swap 2050 MB preserved
c0t2d0s3 /usr 8192 MB
c0t2d0s4 /var 4096 MB
c0t2d0s5 /export 16384 MB
c0t2d0s6 /home 2048 MB
Added Software:
NTP, (Root)
NTP, (Usr)
Removed Software:
Sendmail root
Sendmail user
Region and Locales:
North America
English (United States) (en_US)
English (POSIX C) (C)
System Locale: English (United States,ISO8859-15) (en_US.ISO

Enter 'y' to accept these values and start the installation, or 'n' to return to the beginning and make changes (y/n): y

We are then prompted to insert the software CD's and the OS is installed. At the end of the install we reboot the system. Now that the OS install is complete, and we reboot we are ready to configure the system and start installing software.

Step 3: Basic system changes, updates, and Configuration. Most of this section is adapted from Hal Pomeranz's paper on Hardening Solaris 8.

First, some basic account management changes. We added user accounts for the system administrators, and removed unwanted system accounts: lp, uucp, nuucp, nobody4. Group 30, "admins" was created for administrator accounts. User range 50-99 was chosen for administration accounts who will have access to the whole file system. The range of 100 to 999 has been chosen as internal users, and 1000 to 1999 for customers. Log into the console as root and remove accounts we don't want. Also add an admin (markhamm) and user (hammerm) account.

```
passmgmt -d lp
passmgmt -d listen
passmgmt -d noaccess
passmgmt -d uucp
passmgmt -d nuucp
passmgmt -d nobody4
groupadd -g 30 admins
useradd -u 50  -g 30 -s /bin/sh -c "Mark Admin" -m -d /home/markhamm markhamm
useradd -u 110 -G staff -s /bin/sh -c "Mark User" -m -d /export/jail/./home/hammerm
hammerm
```

Upload the Solaris 9 recommended patch cluster previously downloaded to the workstation, uncompress, install, and reboot. This is important to apply software updates that may address security problems and fix software issues/bugs.

```
unzip 9_Recommended.zip
cd 9_Recommended
./install_cluster
reboot
```

After the reboot, log in as user account to test it, and su to root. There are various system startup scripts and configuration changes we want to make. First check the rc startup folders and disable scripts we don't want running.

In /etc/rc3.d we find startup scripts for NFS and boot server we don't want to run:

```
mv /etc/rc3.d/S15nfs.server /etc/rc3.d/.NOS15nfs.server
mv /etc/rc3.d/S15boot.server /etc/rc3.d/.NOS15boot.server
```

In /etc/rd2.d we find multiple scripts, so we go through them:

S01MOUNTFSYS - keep, We want the file systems to mount at boot.

S05RMTMPFILES - keep, Purging the /tmp files on reboot.

S20sysetup - keep, system scheduling functions.

S30sysid.net - disable, we are replacing with our own script (below)

```
mv /etc/rc2.d/S30sysid.net /etc/rc2.d/.NOS30sysid.net
```

S69inet - keep, this sets up network settings including default route

S71ldap.client - disable, we are not currently using LDAP

```
mv /etc/rc2.d/S71ldap.client /etc/rc2.d/.NOS71ldap.client
```

S71rpc - disable, we do not want RCP services and are not using NIS

```
mv /etc/rc2.d/S71rpc /etc/rc2.d/.NOS71rpc
```

S71sysid.sys - disable, we are done configuring the system and don't need reconfiguration tools running

mv /etc/rc2.d/S71sysid.sys /etc/rc2.d/.NOS71sysid.sys

S72autoinstall - disable, we are done installing and don't want this around any more

mv /etc/rc2.d/S72autoinstall /etc/rc2.d/.NOS72autoinstall

S72inetsvc - disable, we are replacing with our own script (below)

mv /etc/rc2.d/S72inetsvc /etc/rc2.d/.NOS72inetsvc

S73cachefs.daemon - disable, we are not running NFS and don't need caching

mv /etc/rc2.d/S73cachefs.daemon /etc/rc2.d/.NOS73cachefs.daemon

S73nfs.client - disable, we are not using nfs

mv /etc/rc2.d/S73nfs.client /etc/rc2.d/.NOS73nfs.client

S74autofs - disable, we don't need automounting

mv /etc/rc2.d/S74autofs /etc/rc2.d/.NOS74autofs

S74syslog - keep, we want syslog running

S74xntpd - keep, we want to use NTP

S75cron - keep, we want cron running

S75savecore - disable, if we need it to debug problems it should be on the private sytsem

mv /etc/rc2.d/S75savecore /etc/rc2.d/.NOS75savecore

S76nscd - disable, we are not using NIS and don't have need for the caching

mv /etc/rc2.d/S76nscd /etc/rc2.d/.NOS76nscd

S88utmpd - keep, we want the logging / cleanup functionality

S89PRESERVE - disable, we don't want to save temporary editor files

mv /etc/rc2.d/S89PRESERVE /etc/rc2.d/.NOS89PRESERVE

S93cacheos.finish - disable, we are done installing the OS and don't need this around

mv /etc/rc2.d/S93cacheos.finish /etc/rc2.d/.NOS93cacheos.finish

S99audit - keep, we will want to use security auditing

Create a startup script to ensure that all scripts start with proper umask.

```
umask 022  # make sure umask.sh gets created with the proper mode
echo "umask 022" > /etc/init.d/umask.sh
chmod 544 /etc/init.d/umask.sh
chown root:sys /etc/init.d/umask.sh
for d in /etc/rc?.d
do
ln /etc/init.d/umask.sh $d/S00umask.sh
done
```

Create our own network startup script to replace S30sysid.net, S69inet, and S72inetsvc. For now we just want to configure network interfaces and start inetd. Later we will disable inetd and add our own services. Create /etc/init.d/myinetsvc with the following:

```
#!/sbin/sh
```

```
#
# configure network interfaces
/usr/sbin/ifconfig -auD netmask + broadcast +
# Prevent inetd from creating core files
ulimit -c 0
# Run inetd in "standalone" mode (-s) and log all connections (-t)
/usr/sbin/inetd -s -t
```

Set permissions and create a link from rc2.d:

```
chmod 544 /etc/init.d/myinetsvc
chown root:sys /etc/init.d/myinetsvc
ln -s /etc/init.d/myinetsvc /etc/rc2.d/S69myinetsvc
```

We now want to create a startup script to set some parameters using ndd. For more information on ndd settings see: http://www.sun.com/solutions/blueprints/1299/network.pdf. Some of these settings only have an effect if this system is set up as a router, but we chose to add them all to standardise configuration. Create /etc/init.d/nddsettings with the follwing:

```
#!/bin/sh
#
# Fix for broadcast ping bug
/usr/sbin/ndd -set /dev/ip ip_respond_to_echo_broadcast 0
# Block directed broadcast packets.
/usr/sbin/ndd -set /dev/ip ip_forward_directed_broadcasts 0
# Prevent spoofing
/usr/sbin/ndd -set /dev/ip ip_strict_dst_multihoming 1
# Ignore ICMP redirects
/usr/sbin/ndd -set /dev/ip ip_ignore_redirect 1
# No IP forwarding
/usr/sbin/ndd -set /dev/ip ip_forwarding 0
# Drop source routed packets
/usr/sbin/ndd -set /dev/ip ip_forward_src_routed 0
# Shorten ARP expiration to one minute (from 5) to minimize ARP
spoofing/hijacking
/usr/sbin/ndd -set /dev/ip ip_ire_flush_interval 60000
# Shorten the time that routes will be cached from 20 minutes to one
/usr/sbin/ndd -set /dev/arp arp_cleanup_interval 60000
# Disable responding to mask requests, we don't need to send that
information
/usr/sbin/ndd -set /dev/ip ip_respond_to_address_mask_broadcast 0
# Disable responding to ICMP timestamp requests
/usr/sbin/ndd -set /dev/ip ip_respond_to_timestamp 0
# Disable responding to broadcast timestamp request
/usr/sbin/ndd -set /dev/ip ip_respond_to_timestamp_broadcast 0
# Disable send ICMP redirects.  Only needed if is a router
/usr/sbin/ndd -set /dev/ip ip_send_redirects 0
```

 # Disable packets going out a different interface than they arrived
/usr/sbin/ndd -set /dev/ip ip_strict_dst_multihoming 1
 # protect against SYN floods, since the server may get a lot of web traffic
/usr/sbin/ndd -set /dev/tcp tcp_conn_req_max_q0 4096

Set permissions and create a link from rc2.d:

chmod 544 /etc/init.d/nddsettings
chown root:sys /etc/init.d/nddsettings
ln -s /etc/init.d/nddsettings /etc/rc2.d/S70nddsettings

Now that we have disabled unwanted startup scripts disabled, and added our own, we need to make some system configuration changes.

Prevent routing:

touch /etc/notrouter

Modify /etc/default/inetinit to Configure RFC 1948 TCP sequence number generation by setting:

TCP_STRONG_ISS=2

Root should only be able to log into the console, so verify that /etc/default/login has CONSOLE set:

CONSOLE=/dev/console

We also want to restrict who can run su. We are going to do this by creating a suers group and will add users who need su to that group:

/usr/sbin/groupadd -g 13 suers
/usr/bin/chgrp suers /usr/bin/su /sbin/su.static
/usr/bin/chmod 4550 /usr/bin/su /sbin/su.static

We want to add some logging, and send the logs from the public system to the private one. Syslog is a topic of its own, and is well covered in other documents. The private system is configured as the syslog server, so here we are only covering the settings for the public system. The following lines are added to /etc/syslog.cnf to do some local logging, and send all logging information to the log server:

```
mail.debug    /var/adm/maillog
daemon.info   /var/adm/daemonlog
*.debug       @logger.myorg.org
```

Next we want to set up NTP. There is already an NTP server on the network, so we need to configure to use that. We create a ntp.conf file in /etc/inet:

```
restrict default ignore
restrict 192.168.0.0 mask 255.255.255.0 noquery nomodify notrap
restrict 127.0.0.1
```

```
broadcastclient key 1
driftfile       /var/ntp/ntp.drift
enable          auth
keys            /etc/inet/ntp.keys
trustedkey      1
```

Then we set up the ntp.key file also in /etc/inet. It's just a one line file, with the password from the NTP server.

```
1       M       passcodehere
```

Kernel Parameters:
The /etc/system file was edited to add the following settings against some attacks.

```
# protect against stack "smashing" / overruns
set noexec_user_stack = 1
set noexec_user_stack_log = 1
# limit processes per user to 128 (instead of the default 7877)
set maxuprc = 128
# disable core dumps.  no development is being done on this system
set sys:coredumpsize = 0
```

Drive mount settings:
Now we want to enable some settings on the drives. We want to enable logging on all of the mount points to recover in case of a crash. We also want to mount var, export, and home with nosuid. So we edit /etc/vfstab to add these settings.

```
#device        device          mount           FS      fsck    mount   mount
#to mount      to fsck         point           type    pass    at boot options
#
fd      -       /dev/fd fd      -       no      -
/proc   -       /proc   proc    -       no      -
/dev/dsk/c0t2d0s1       -       -       swap    -       no      nosuid
/dev/dsk/c0t2d0s0       /dev/rdsk/c0t2d0s0      /       ufs     1       no      logging
/dev/dsk/c0t2d0s3       /dev/rdsk/c0t2d0s3      /usr    ufs     1       no      logging
/dev/dsk/c0t2d0s4       /dev/rdsk/c0t2d0s4      /var    ufs     1       no      logging,
nosuid
/dev/dsk/c0t2d0s5       /dev/rdsk/c0t2d0s5      /export/        ufs     2       yes     logging,
nosuid
/dev/dsk/c0t2d0s6       /dev/rdsk/c0t2d0s6      /home/  ufs     2       yes     logging,
nosuid
```

Default password settings:
We set some default password settings. The minimum length of 6 is ok, but we also want to make sure the password aging is enabled. Not all of the password settings we want can be defaulted, but if the administrator forgets to set them when creating the account they will have some reasonable numbers. Solaris 9

also does some "easy" password protection with passwd, requiring a non-alpha character and requiring something unlike the account name, and having more than 2 characters different when changing password. Given that, we set the MAXWEEKS and MINWEEKS in /etc/default/passwd:

```
MAXWEEKS=13
MINWEEKS=1
PASSLENGTH=6
```

Enable Auditing:

1. Enable the Basic Security Module (BSM)
   `/etc/security/bsmconv`
2. Configure the classes of events to log in /etc/security/audit_control:

```
dir:/var/audit
flags:lo,ad,pc,fc,fd,fm
naflags:lo,ad
#
#   lo - login/logout events
#   ad - administrative actions: mount, exportfs, etc.
#   pc - process operations: fork, exec, exit, etc.
#   fc - file creation
#   fd - file deletion
#   fm - change of object attributes: chown, flock, etc.
#
```

3. Create /etc/security/newauditlog.sh:

```
#!/sbin/sh
#
# newauditlog.sh - Start a new audit file and expire the old logs
#
AUDIT_EXPIRE=30
AUDIT_DIR="/var/audit"
/usr/sbin/audit -n
cd $AUDIT_DIR # in case it is a link
/usr/bin/find . $AUDIT_DIR -type f -mtime +$AUDIT_EXPIRE \
   -exec rm {} > /dev/null 2>&1 \;
```

4. Run the script nightly from cron:
   `chmod 500 /etc/security/newauditlog.sh`
   `/usr/bin/crontab -e root`
   `0 0 * * * /etc/security/newauditlog.sh`
5. The audit files should be backed up to tape, and randomly reviewed.

Last we obtained fix-modes script from: http://www.sun.com/solutions/blueprints/tools/. We then compiled it on the private system, transferred "secure-modes" and script "fix-modes" to the same directory on the public system.

```
tar -xvf FixModes.tar
cd FixModes
```

make
scp fix-modes hammersm@public.myorg.org:fix-modes
scp secure-modes hammersm@public.myorg.org:secure-modes

We then ran ran "fix-modes" on the public system, and re-checked for suid files to obtain an updated baseline. At this point the OS is considered reasonably secure.

Stage Two: Install Applications

Now we are ready to install the software chosen to provide the necessary services. Apache 2 was chosen as the web server. OpenSSH was chosen to provide the ability to connect to the server and to be used for uploading/downloading files. Sendmail 8.12 was chosen to handle the mail services, and ISC BIND for the DNS services. Zlib and SSL are necessary prerequisites. Even though there are only two people who need root, SUDO was also installed for better control and logging.

Install Zlib:
Zlib is required so we need to install it in the default location /usr/lib.

tar -xvf zlib-1.1.4.tar
cd zlib-1.1.4
./configure
make
make install

Install SSL/SSH:
Used a modified session.c in SSH to enable CHROOT on a user by user basis. This is adapted from a document posted by Kent Cowgill (kent@c2group.net).

Install Openssl: uncompress the source tar, install using defaults

tar -xvf openssl-0.9.7b.tar
cd openssl-0.9.7b
./config
make
make install

To use privilege separation (now default in openSSH), we need to create the sshd user and group (if they don't already exist). I picked user/group 22 because they are the port for SSH and happen to be available. The Group and User IDs needs to be the same both systems:

mkdir /var/empty
chown root:sys /var/empty
chmod 755 /var/empty

groupadd -g 22 sshd
passmgmt -a -c "sshd privsep" -h /var/empty -u 22 -g 22 sshd

Install Openssh: Uncompress the source tar, install with PAM support. We considered not adding PAM support as it removed the need to copy the /usr/lib/security files into the CHROOT environment, but decided it the PAM support would be preferable.

tar -xvf openssh-3.6.1p2.tar
cd openssh-3.6.1p2
./configure --with-pam --with-ipv4-default

Before compiling and installing SSH, we need to apply the "chroot patch". I obtained this from http://chrootssh.sourceforge.net/ (See Appendix A). Save (or create) the file /tmp/chroot-session.patch. The functionality of the patch is fairly simple. It looks for a "/./" in the users home directory, and if it finds it, chroots to what was before the /./, and the users "new" home directory becomes what was after the "/./".

patch -p0 < /tmp/chroot-session.patch

Now that the patch is applied, we are ready to install.

make
make install

At the end of the install we need to set up a start up script for SSH. We copied the opensshd.in provided in contrib/solaris of openssh-3.6p1 to /etc/init.d, and then made a link in rc2.d to launch it on startup.

cp contrib/solaris/opensshd.in /etc/init.d/opensshd.in
ln -s /etc/init.d/opensshd.in /etc/rc2.d/S75opensshd

Configuration changes to SSH: Before starting SSH we want to make some changes to the configuration. Edit the /usr/local/etc/sshd_config file. We only want to allow SSH protocol 2, so uncomment the "Protocol 1 2", and change it to read:

Protocol 2

Also verify that the setting "PermitRootLogin" is set to no. We do not want to allow root to connect directly through SSH. Then go to the "Banner", uncomment it, and set it to:

Banner /etc/default/sshd

Exit and save the sshd_config. Now we want to create the sshd default banner fille:

echo "Authorized uses only. All access may be logged." > /etc/default/sshd
chmod 444 /etc/default/sshd
chown root:sys /etc/default/sshd

With Libz, SSL & SSH installed and configured on the private system, we now want to tar it up and move it to the public system. The best way to do this is tar up what we want from the private system and transfer it over.

```
tar -cvf /tmp/ssh.tar /usr/local/ssl
tar -cvf /tmp/ssh.tar /usr/lib/libz.a
tar -cvf /tmp/ssh.tar /usr/lib/libz.so.1
tar -uvf /tmp/ssh.tar /usr/local/bin/scp
tar -uvf /tmp/ssh.tar /usr/local/bin/sftp
tar -uvf /tmp/ssh.tar /usr/local/bin/slogin
tar -uvf /tmp/ssh.tar /usr/local/bin/ssh
tar -uvf /tmp/ssh.tar /usr/local/bin/ssh-add
tar -uvf /tmp/ssh.tar /usr/local/bin/ssh-agent
tar -uvf /tmp/ssh.tar /usr/local/bin/ssh-keygen
tar -uvf /tmp/ssh.tar /usr/local/bin/ssh-keyscan
tar -uvf /tmp/ssh.tar /usr/local/etc/ssh_config
tar -uvf /tmp/ssh.tar /usr/local/etc/sshd_config
tar -uvf /etc/init.d/opensshd.in
tar -uvf /etc/default/sshd
```

Now transfer the /tmp/ssh.tar over, and untar it onto the public system. We then create the startup link to start sshd, and generate the ssh keys:

```
cd /
tar -xvf /tmp/ssh.tar
ln -s /etc/init.d/opensshd.in /etc/rc2.d/S75opensshd
/usr/local/bin/ssh-keygen -t rsa1 -f /usr/local/etc/ssh/ssh_host_key -N ""
/usr/local/bin/ssh-keygen -t rsa -f /usr/local/etc/ssh/ssh_host_rsa_key -N ""
/usr/local/bin/ssh-keygen -t dsa -f /usr/local/etc/ssh/ssh_host_dsa_key -N
""
```

Now SSH is installed and ready to use on both systems, reboot to have all the configurations applied and make sure SSH starts.

```
reboot
```

When the systems are done booting, and attempt to connect via SSH. If both are working properly we are ready to disable INETD. Edit the /etc/init.d/myinetsvc, and coment inetd:

```
#/usr/sbin/inetd -s -t
```

We can now kill inetd since we no longer need it.

Now we are ready to set up the CHROOT jail for the restricted users. The following will give our users the tools they need to transfer files and move around in their jail. Some of the tools to do this (such as LDD) are not available on the public system, so we go back to the private system to build the jail structure.

```
#First build the Directory Structure
```

```
JAIL=/export/buildjail
export JAIL
cd $JAIL
mkdir etc
mkdir usr
mkdir usr/bin
mkdir usr/local
mkdir usr/local/bin
mkdir usr/local/libexec
mkdir usr/local/sbin
mkdir usr/local/lib
mkdir usr/local/ssl
mkdir usr/local/ssl/lib
mkdir usr/lib
mkdir usr/platform
mkdir usr/platform/`uname -i`
mkdir usr/platform/`uname -i`/lib
mkdir dev
mkdir devices
mkdir devices/pseudo
mkdir home
mkdir var
mkdir var/tmp
mkdir var/empty
ln -s ./usr/lib lib
ln -s ./usr/bin bin
ln -s ./var/tmp tmp

# copy over necessary binaries and libraries
APPS='bin/cp bin/ls bin/mkdir bin/mv bin/pwd bin/rm bin/rmdir bin/sh'
for i in $APPS; do
  cp /$i ./$i
  LIBS=`ldd ./$i | awk '{print $3}'`
  for l in $LIBS; do
    mkdir ./`dirname $l` > /dev/null
    cp $l .$l
  done
done

# additional required binaries and libraries:
cd $JAIL
BINS="usr/local/bin/ssh usr/local/libexec/sftp-server usr/local/sbin/sshd
usr/lib/libz.so usr/local/ssl/lib/libcrypto.a usr/lib/ld.so.1 usr/platform/`uname
-i`/lib/libc_psr.so.1 usr/lib/nss_files.so.1"
for i in $BINS; do
  cp /$i ./$i
```

```
done

# create necessary devices
cd $JAIL/devices/pseudo
mknod mm@0:zero c 13 12
mknod mm@0:null c 13 2
cd $JAIL/dev
ln -s ../devices/psuedo/mm@0:zero zero
ln -s ../devices/pseudo/mm@0:null null
cp -Rp /dev/random .
cp -Rp /dev/urandom .

# tar up the JAIL so it is ready to move over to the public system
cd $JAIL
tar -cvf /tmp/basejail.tar *
```

Now that we have the structure we want for the CHROOT jail. We become root on the private system, transfer the tar file over, and unpack it in our jail. We put the tar file in /opt since it can be used to easily build other CHROOT areas. The jail area will be /export/jail:

```
/usr/local/bin/scp hammersm@192.168.0.20:/tmp/basejail.tar
/opt/basejail.tar
mkdir /export/jail
cd /export/jail
tar -xvf /opt/basejail.tar
```

What we still need in the jail are password and group files. The password file needs to contain at least the users that will be using that jail. The group can be empty, but it is easier on the users if the group-owners show up by name. This script will create a modified password file in the jail with just root and the chrooted users, and show the virtual home folder. This will need to be run any time a "jail" user is added to the system.

```
#!/bin/sh
# Put users into the chrooted password file, strip off the leading
JAIL=/export/jail; export JAIL
egrep '\/\.\/|\:0\:' /etc/passwd |sed -e 's/\:\/[a-zA-Z0-9\/]*\./:/' >
$JAIL/etc/passwd
cp /etc/group $JAIL/etc/group
```

Install Sendmail:

Solaris 9 comes with Sendmail 8.12, however we decided to install the open source version. The server is primarily acting as a gateway, and the virtualusertable and aliases will be used to route mail to appropriate users. On the private system we unpacked the sendmail-8.12.9 file, and then set up the sendmail.mc file.

```
tar -xvf sendmail.8.12.9.tar
cd sendmail.8.12.9
cd cf/cf/
cp generic-solaris.mc sendmail.mc
```

We then modified the sendmail.mc to use the settings we want. We want mail going out from the sytem to come from myorg.org instead of thisserver.myorg.org so used the masquerade setting. We needed virtualusertable to route mail based on domains, and we want to use the access features. We also added no_default_msa to disable sendmail listening on port 587, and confPRIVACY_FLAGS=goaway to disable EXPN and VRFY commands. We also set the Logon message to less information.

```
divert(0)dnl
VERSIONID(`$Id: generic-solaris.mc,v 8.13 2001/06/27 21:46:30 gshapiro
Exp $')
OSTYPE(solaris2)dnl
DOMAIN(generic)dnl
FEATURE(`masquerade_envelope')dnl
MASQUERADE_AS(`myorg.org')dnl
MASQUERADE_DOMAIN(`localhost')dnl
EXPOSED_USER(`root')dnl
FEATURE(`use_cw_file')dnl
FEATURE(`use_ct_file')dnl
FEATURE(`virtusertable', `dbm /etc/mail/virtusertable')dnl
FEATURE(`access_db')dnl
FEATURE(`no_default_msa')dnl
define(`confPRIVACY_FLAGS', `goaway')dnl
define(`ALIAS_FILE', `/etc/mail/aliases')dnl
define(`confSMTP_LOGIN_MSG', `MyOrg SMTP Gateway')dnl
MAILER(local)dnl
MAILER(smtp)dnl
```

Now with the sendmail.mc file created, we create and install the sendmail.cf.

```
sh Build sendmail.cf
sh Build install-cf
```

Since we are installing on Solaris 9 the smmsp user and group used by sendmail are already on the system. Now we just need to build and install it. The man pages caused an error unless we created the folders first.

```
cd ../../sendmail
sh Build
mkdir /usr/share/man/cat8/
mkdir /usr/share/man/cat5/
mkdir /usr/share/man/cat1/
sh Build install
```

Now that sendmail is installed on the private system we need to copy it over to the public system. The following creates a tar of all the files we needed.

```
tar -cvf /tmp/sm.tar /etc/mail/aliases
tar -uvf /tmp/sm.tar /etc/mail/access
tar -uvf /tmp/sm.tar /etc/mail/local-host-names
tar -uvf /tmp/sm.tar /etc/mail/sendmail.cR
tar -uvf /tmp/sm.tar /etc/mail/sendmail.cf
tar -uvf /tmp/sm.tar /etc/mail/sendmail.hf
tar -uvf /tmp/sm.tar /etc/mail/sendmail.st
tar -uvf /tmp/sm.tar /etc/mail/trusted-users
tar -uvf /tmp/sm.tar /etc/mail/virtusertable
tar -uvf /tmp/sm.tar /etc/mail/mailx.rc
tar -uvf /tmp/sm.tar /usr/lib/mail.local
tar -uvf /tmp/sm.tar /usr/lib/sendmail
tar -uvf /tmp/sm.tar /usr/lib/smrsh
tar -uvf /tmp/sm.tar /etc/init.d/sendmail
tar -uvf /tmp/sm.tar /usr/sbin/makemap
tar -uvf /tmp/sm.tar /usr/sbin/mailstats
tar -uvf /tmp/sm.tar /usr/bin/purgestat
tar -uvf /tmp/sm.tar /usr/bin/newaliases
tar -uvf /tmp/sm.tar /usr/bin/mailq
tar -uvf /tmp/sm.tar /usr/bin/hoststat
tar -uvf /tmp/sm.tar /etc/init.d/sendmail
```

Then we transfer the tar file over to the public system and un-tar it. Once that is done we need to create the mail queue folder since we didn't run the install scrip on the public sytem.

```
mkdir /var/spool/mqueue
chown root:bin /var/spool/mqueue
chmod 750 /var/spool/mqueue
```

Before starting sendmail we need to generate the aliases, acess, and virtualusertable database files. The necessary entries in aliases and virtusertable for routing mail to the internal mail servers are set up first, and we add some blocks to the access file. We also add "192.168.0 RELAY" to access to allow the internal servers to route mail out. The firewall only allows traffic from 192.168.0.0/16 to come from the internal network, so this wont allow us to be an open relay.

```
cd /etc/mail
newaliases
makemap access<access
makemap virtusertable<virtusertable
```

Now we are ready to start sendmail, and set it to autostart on reboot.

```
/etc/init.d/sendmail start
ln -s /etc/init.d/sendmail  /etc/rc2.d/S88sendmail
```

Install Apache:

We intended to installed Apache in a chroot jail, but due to restricted timeline the initial install is in /usr/local/apache2. First we build it and install it ion the private system, then tar it up to move over to the public sytem.

```
./configure --prefix=/usr/local/apache2 --enable-ssl --with-ssl=/usr/local/ssl
--enable-rewrite
make CC=cc
make install
tar -cvf /tmp/apache.tar /usr/local/apache2
```

Once the install is done, we move the tar file over to the public system and untar it.

Since we want SSL available, we need a server certificate. SSL is currently only going to be used in house so a self signed certificate will be used. In the future a certificate using a root authority can replace it. This was done in the apache config directory.

```
cd /usr/local/apache2/conf
```

First we need to generate a certificate request and a private key. This generates several prompts: passphrase, Country, State, City, Organization, Organizational Unit, a Common name, and an Email contact. One thing to not is the common name needs to be the name users will access the server as, such as "www.myorg.org".

```
/usr/local/ssl/bin/openssl req -new > my.csr
```

Now that we have a certificate request, my.csr, and our private encrypted key, privkey.pem. We want a non-encrypted private key, server.key, so Apache will auto start without prompting for a password.

```
/usr/local/ssl/bin/openssl rsa -in privkey.pem -out server.key
```

This will need passphrase to decrypt the privkey.pem to server.key. Now we have a server.key to use to start the webserver. Last we need to generate a public certifiate to us with the web server. Here we generate a my.cert that will be valid for 999 days.

```
/usr/local/ssl/bin/openssl x509 -in my.csr -out my.cert -req -signkey my.key
-days 999
```

The server.key and privkey.pem need to be protected. If someone does acquire these they can pretend to be our server. Since apache2 defaults to using ssl.key/server.key we are going to put the files there.

```
mkdir ssl.key
chown root:sys ssl.key
chmod 500 ssl.key
```

```
mv server.key ssl.key
mv privkey.pem ssl.key
```

The public certificate needs to be in an accessable directory, so we also put it in a default directory for Apache 2.

```
mkdir ssl.crt
mv my.cert ssl.crt/server.crt
```

Now we want a www user/group for apache to run as:

```
groupadd -g 80 www
useradd -u 80 -g 80 -s /bin/sh -c "Web Server" -d /usr/local/apache2 www
```

Then edit the httpd.conf file and make some changes:

```
# set user & group so apache runs as www.
User www
Group www
# set ServerTokens Prod to limit software information given out by apache.
ServerTokens Prod
#  In the global options section, we want to turn off showing directory indexes
# so we add -Indexes to the Options
   Options FollowSymLinks -Indexes
# we also add a redirect so if users attempt to access our password change page via HTTP it sends them to HTTPS.
Redirect permanent /cgi-bin/changepw.pl https://www.myorg.org/cgi-bin/changepw.pl
```

The next group of settings are modified in the httpd.conf for non-secure browsing, but need to also be modified in the virtual host in the ssl.conf file to match.

```
# set an email contact
ServerAdmin webadmin@myorg.org
# change the document root to the webdata folder in the choroot environment.
DocumentRoot "/export/jail/webdata"
```

Now we can start apache and make sure it is working:

```
../bin/apachectl startssl
```

For security reasons we chose not to have apache auto start on reboot, therefore are not creating any rc startup scripts for it. It is also policy that any cgi files need to be reviewed by administration before being put into production. This prevents any potentially damaging cgi files from being put in place without the administrators knowledge.

Install Perl Modules:

We decided to provide a Web page to allow the web developers to change their passwords. This is in part to make it easier for the user to do so, but also to get around them not having access to the shadow in the chroot environment. To get the script to work, we installed the perl modules Expect-1.15, IO-Stty-.02, IO-Tty-1.02, and Syslog-0.97. We used the default install for the modules, which updated and/or added files to various locations under the /usr/perl5 directory. After installing the modules on the private system, the entire /usr/perl5 directory was copied over to the public system to make the modules available on it.

Install Web-Based change password tool:

The perl program to change the password has two parts. Using Expect allows this program to run without ever needing "root". We split the program up using perl sockets so it will still function in the future if Apache is run in a chroot environment. The user iterface part of the script we called pw1.pl, and the daemon script is pwd.pl.

Using a rewrite rule in apache access to the user interface script is forced to use https to prevent passwords going over the net in clear text. This script takes the users login name, old password, and new password, and verification of the new password. That information is then passed through a perl socket to the daemon process. The daemon then uses Expect to run su with passwd as the shell to change the users password. Since the pwd.pl script uses su, we need to make sure www can run su by adding www to the suers group.

Security precautions:

- In the input script (pw1.pl) basic checks are done to see that the input seems valid.
- The socket file used to communicate between the two scripts is accessable only by the www user.
- The daemon script (pwd.pl) runs as www, and never has root privelages. It does some validation on the input data before using su and passwd to change the users password. Any detected hack attempts will cause the daemon to abort and log why.
- In Apache added the redirect so request to the password change page are encrypted:
  Redirect permanent /cgi-bin/pw1.pl https://public.myorg.org/cgi-bin/pw1.pl

Install BIND:

Since this server is also going to act as our DNS server, se we need to install BIND on it. First we extract and build it on the private system. We want to put the files in their own structure so specify a prefix, and when are making ssl

avaialble to use if we use it for a backup name server later. We then tar up the bind folder to move to the public system.

```
tar -xvf bind-9.2.2.tar
cd bind-9.2.2
./configure --prefix=/usr/local/bind --with-openssl
make
make install
tar -cvf /tmp/bind.tar /usr/local/bind
```

Once we copy the tar over to the public system and extract it we need to set up Bind. We set up the named.conf file and the zone files. We choose to keep the configuration files for bind in /usr/local/bind/conf.

To prevent our name server from being used by everyone, in the named.conf global options section we set up an allow-querry group listing only our internal networks. To allow global access to lookups for our servers, "allow-query { any; };" is added for the specific zones we host DNS for.

Once the files are updated with the DNS entries we want, we create a link from etc to the our named.conf file from the /etc folder:

```
ln -s /usr/local/bind/conf/named.conf /etc/named.conf
```

We also need to create the db.cache file. The current version can be obtained at ftp://ftp.rs.internic.net/domain/db.cache.

Next we need to configure the system to resolve using the local name service by editing the /etc/resolv.conf and making the first name server 127.0.0.1.

Then we can start BIND and test it.

```
/usr/local/bind/sbin/named
```

To have BIND start at system startup, we edit our /etc/init.d/myinetsvc and add:

```
#start BIND
/usr/local/bind/sbin/named
```

Install Sudo:

We want to use sudo to help track su actions as well as the potential to give some users access to root functions in the future. The first option chosen was to have mail sent to the two admins at the internal mail servers. Additionally with-logging=both allows logs to both syslog and the sudo.log file. Having the "." in the path is against good practice, so using with-ignore-dot to force it. Prevent root from running sudo with disable-root-sudo. Last we add with-all-insults for the entertainment.

```
tar -xvf sudo-1.6.6.tar
```

```
./configure --with-
mailto=admin1@mail1.myorg.org,admin2@mail2.myorg.org \
--with-logging=both --with-logpath=/var/adm/sudo.log \
--with-ignore-dot --disable-root-sudo --with-all-insults
make
make install
```

We then need to set up the sudo file. Currently it only has the two administrators in it, and they need to have full access. To do that, they are given access to all commands. Granted, this does make it possible for the administrator to sudo to a root shell, but it is against policy, and removes the overhead of maintaing a list of sudo commands. We then copy sudo over to the public system.

```
tar -cvf /tmp/sudo.tar /usr/local/bin/sudo
tar -uvf /tmp/sudo.tar /usr/local/sbin/visudo
tar -uvf /tmp/sudo.tar /usr/local/etc/sudoers
scp /tmp/sudo.tar hammersm@public.myorg.org:/tmp/sudo.tar
```

Back everything up:

Now that the system is ready before we put it into place, we need to make a backup. Since the system does not have much data, the /tmp volume has enough room. Boot to single user mode and log in as root, and tar up the files. We then burn this tar to a CD.

```
tar -cvf /tmp/backup.tar /etc
tar -uvf /tmp/backup.tar /export
tar -uvf /tmp/backup.tar /home
tar -uvf /tmp/backup.tar /kernel
tar -uvf /tmp/backup.tar /opt
tar -uvf /tmp/backup.tar /platform
tar -uvf /tmp/backup.tar /sbin
tar -uvf /tmp/backup.tar /usr
tar -uvf /tmp/backup.tar /var
```

Maintence:

Monitoring:

Automated monitoring is done from another system, logger.myorg.org This system is running several apps to monitor the other systems, as well as acting as the central log server.

Nagios: Nagios is running a check every 5 minutes that SSH, DNS, HTTP, HTTPS, and SMTP are responding. Failure to respond triggers an email alert. Continued failure of the service will trigger a pager message after 30 minutes.

ChkProc: A simple script to check to verify processes are running. It checks to see if syslog, BIND, Apache, and sendmail are running. If they are not it attempts to restart the process, and echo's a message that is emailed to the administrators.

Logcheck: Log file checker (logcheck.sh) written by Craig Rowland <crowland@psionic.com> is installed on the monitoring system to monitor the logs. It sends an email every 5 minutes with suspicious messages from the logs to the administrators.

Patching & Updates:

Patching with the Solaris recommended is scheduled to occur on the private system the 3'rd Friday, and if no problems arise, then to the public server a week later.

Installed software is scheduled to be reviewed quarterly to see if there are useful updates or changes that should be made.

Other patches may be applied as deemed necessary. See Security Alerts below.

Security Alterts:

Security alert bulletins, including the CERT advisory's and SANS bulletins will be monitored for critical patches or security problems that may come up. These advisories are likely to cause patching outside the scheduled patch reviews.

Log rotation:

The Security Audit logs are archived to CD weekly, and are randomly reviewed for strange behavior.

The syslog logs are scheduled to be rotate weekly, with 8 rotations kept local then purged after 8 weeks using a script from the Sans Solaris Security guide (Appendix E). Permanent copies of the logs are archived off of the main logging server monthly.

The Apache log is rotated monthly. The log file is then zipped and made available via SCP to users that want to review it. See Appendix F for the script.

Connection Monitoring:

Monitor the firewall for any outgoing connections from the system. The only valid new outgoing connections that should show up are DNS. Any other traffic will be investigated.

User Accounts:

User accounts should be reviewed monthly. Accounts are created with the following settings:

- set the password to expire after 90 days.
- require 3 days between password changes
- warn the user 14 days before the password expires.
- set account disable after 91 days of incativity

Users should be reviewed monthly to verify that a user wasn't created without these settings. All accounts should be reviewed to verify that they should still be on the system, and if not the account is removed.

Suid files:

Monthly a list of the suid files is generated and compared to the master list to verify that no new suid files have show up.

Backups:

Part of any system maintenance should be a backup. Since there isn't currently the hardware available to back the system up locally, the policy is to make a backup of any configuration file to the private system when changes are made. The private system is then backed up on a regular basis to tape. The website content mangers are told to back up their own content, but it is also backed up weekly by transferring the site to the private system using SCP.

Future planning:

Items that were considered but not done at this time but considered, and possibly will be implemented in the future.

-Build a Sudo config with sets of commands associated with duties so that other staff can assist in administration of certan areas of the server, such as the Apache server.
-Install BIND into a CHROOT environment
-Install Apache into a CHROOT environment
-Rearrange some file locations and mount points be about to mount some RO. Due to the default and/or chosen locations for files, none of the mount points on this system can be mounted read only.
-Install Tripewire, or build/find another similar solution to monitor changes in files.

Verify Security Configuration:

SUID files:

Find suid files and review them. The follwing command will show all suid files on the system. The list of files was reviewed, and stored. This is done monthly as part of maintenance and compared to see if any new suid files show up.

```
/bin/find / -type f \( -perm -4000 -o -perm -2000 \) -exec ls -ldb {} \;
```

To double verify that no SUID files are in the /export folder that can be accessed by the chroot users:

```
/bin/find /export -type f \( -perm -4000 -o -perm -2000 \) -exec ls -ldb {} \;
```

Netstat: The first simple check is to run netstat -an to identify what ports we are listening on:

```
# netstat -an

UDP: IPv4
   Local Address        Remote Address     State
-------------------- -------------------- -------
      *.123                                Idle
127.0.0.1.123                              Idle
192.168.0.19.123                           Idle
      *.123                                Idle
      *.*                                  Unbound
      *.514                                Idle
127.0.0.1.53                               Idle
192.168.0.19.53                            Idle
      *.32872                              Idle
      *.32873                              Idle
      *.*                                  Unbound

UDP: IPv6
   Local Address                 Remote Address                State      If
-------------------------------- -------------------------------- ---------- -----
      *.32873                                                   Idle

TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
      *.*                  *.*              0      0 49152      0 IDLE
      *.22                 *.*              0      0 49152      0 LISTEN
192.168.0.22     192.168.0.33.2230   64807      0 49884     52 ESTABLISHED
```

```
      *.80               *.*              0      0 49152      0 LISTEN
      *.443              *.*              0      0 49152      0 LISTEN
      *.25               *.*              0      0 49152      0 LISTEN
127.0.0.1.53             *.*              0      0 49152      0 LISTEN
192.168.0.19.53          *.*              0      0 49152      0 LISTEN
      *.*                *.*              0      0 49152      0 IDLE

TCP: IPv6
   Local Address                 Remote Address              Swind Send-Q Rwind
Recv-Q   State      If
-------------------------------- -------------------------------- ----- ------ ----- ------ ----------- ---
--
      *.*                        *.*                        0      0 49152      0 IDLE
      *.80                       *.*                        0      0 49152      0 LISTEN
      *.443                      *.*                        0      0 49152      0 LISTEN
```

According to Netstat we are only listening on the ports we intend to. Now we run a Nmap scan against the system to verify:

Run Nmap:

```
# nmap -v -P0 -O 192.168.0.19
No tcp, udp, or ICMP scantype specified, assuming SYN Stealth scan. Use -sP if
you really don't want to portscan (and just want to see what hosts are up).

Starting nmap 3.30 ( http://www.insecure.org/nmap/ ) at 2003-08-08 13:58 PDT
Host public.myorg.org (192.168.0.19) appears to be up ... good.
Initiating SYN Stealth Scan against publid.myorg.org (192.168.0.19) at 13:58
Adding open port 25/tcp
Adding open port 53/tcp
Adding open port 443/tcp
Adding open port 22/tcp
Adding open port 80/tcp
The SYN Stealth Scan took 313 seconds to scan 1644 ports.
For OSScan assuming that port 22 is open and port 1 is closed and neither are
firewalled
Interesting ports on public.myorg.org (192.168.0.19):
(The 1639 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
25/tcp     open        smtp
53/tcp     open        domain
80/tcp     open        http
443/tcp    open        https
Device type: general purpose
Running: Sun Solaris 9
```

OS details: Sun Solaris 9 with TCP_STRONG_ISS set to 2
Uptime 0.071 days (since Fri Aug 8 12:21:34 2003)
TCP Sequence Prediction: Class=truly random
Difficulty=9999999 (Good luck!)
IPID Sequence Generation: Incremental
Nmap run completed -- 1 IP address (1 host up) scanned in 323.254 seconds

Run Nessus:

Run Nessus to look for vulnerabilitieis / possibly overlooked issues. The header for the results was:

Nessus Scan Report

SUMMARY
- Number of hosts which were alive during the test : 1
- Number of security holes found : 0
- Number of security warnings found : 7
- Number of security notes found : 23

TESTED HOSTS
192.168.0.19 (Security warnings found)

DETAILS
+ 192.168.0.19 :
. List of open ports :
  o ssh (22/tcp) (Security warnings found)
  o smtp (25/tcp) (Security notes found)
  o domain (53/tcp) (Security warnings found)
  o http (80/tcp) (Security warnings found)
  o https (443/tcp) (Security warnings found)
  o ntp (123/udp) (Security warnings found)
  o domain (53/udp) (Security notes found)
  o general/udp (Security notes found)
  o general/tcp (Security warnings found)
  o general/icmp (Security warnings found)

All of the results were investigated.

SMTP: No warnings, only expected information was reported.
DNS: Since it was scanned from another internal system recursive queries could be run, this was expected.
HTTP/HTTPS: Found to be vulnerable to cross-site-scripting. Due to the low level of vulnerability, we chose not to install the Rewrite module for Apache and add the overhead of a "fix".
HTTPS: 2 "weak export class" ciphers were found. These may be necessary for exported browsers to use SSL, so they were left as an option.

TCP: Warning about accepting TCP SYN packets with FIN flag set. We are up to current patch, so this is not fixed. However, these are not a concern because they are blocked by the firewall.
NTP: Warning is a default because NTP is running. NTP from all but the NTP servers will be blocked by the firewall.
ICMP: The ICMP responded to a mask request. The risk from this is low (they can get a respons of what net mask is configured), and there doesn't appear to be a way to disable this in Solaris.

Process Check: Check the running processes to verify that there are not any processes we do not expect to be running:

```
# ps -aef
     UID   PID  PPID  C    STIME TTY      TIME CMD
    root     0     0  0   Aug 11 ?       0:04 sched
    root     1     0  0   Aug 11 ?       0:00 /etc/init -
    root     2     0  0   Aug 11 ?       0:00 pageout
    root     3     0  0   Aug 11 ?       2:20 fsflush
    root   165     1  0   Aug 11 ?       0:00 /usr/lib/saf/sac -t 300
    root   168   165  0   Aug 11 ?       0:00 /usr/lib/saf/ttymon
    root   780   708  0 11:14:41 ?       0:00 ./sshd
    root    50     1  0   Aug 11 ?       0:00 /usr/lib/sysevent/syseventd
    root    58     1  0   Aug 11 ?       0:00 /usr/lib/picl/picld
    root   146     1  0   Aug 11 ?       0:00 /usr/sbin/cron
     www   800   799  0 11:15:19 ?       0:00 /usr/local/apache2/bin/httpd -k start -
DSSL
    root   143     1  0   Aug 11 ?       0:00 /usr/sbin/syslogd
    root   145     1  0   Aug 11 ?       0:04 /usr/lib/inet/xntpd
    root   166     1  0   Aug 11 console  0:00 /usr/lib/saf/ttymon -g -h -p rodents
console login:  -T sun -d /dev/console -l
    root   158     1  0   Aug 11 ?       0:00 /usr/lib/utmpd
markhamm  782   780  0 11:14:45 ?       0:00 ./sshd
markhamm   784   782  0 11:14:45 pts/1    0:00 -sh
    root   788   784  0 11:14:51 pts/1    0:00 sh
    root   794     1  0 11:15:04 ?       0:00 /usr/lib/sendmail -bd -q15m
    root   808     1  0 11:15:38 ?       0:00 /usr/local/bind/sbin/named
    root   809   788  0 11:15:42 pts/1    0:00 ps -aef
     www   801   799  0 11:15:19 ?       0:00 /usr/local/apache2/bin/httpd -k start -
DSSL
     www   803   799  0 11:15:19 ?       0:00 /usr/local/apache2/bin/httpd -k start -
DSSL
     www   804   799  0 11:15:19 ?       0:00 /usr/local/apache2/bin/httpd -k start -
DSSL
    root   708     1  0 13:35:13 ?       0:00 ./sshd
     www   802   799  0 11:15:19 ?       0:00 /usr/local/apache2/bin/httpd -k start -
DSSL
```

root 799 1 2 11:15:18 ? 0:01 /usr/local/apache2/bin/httpd -k start -DSSL

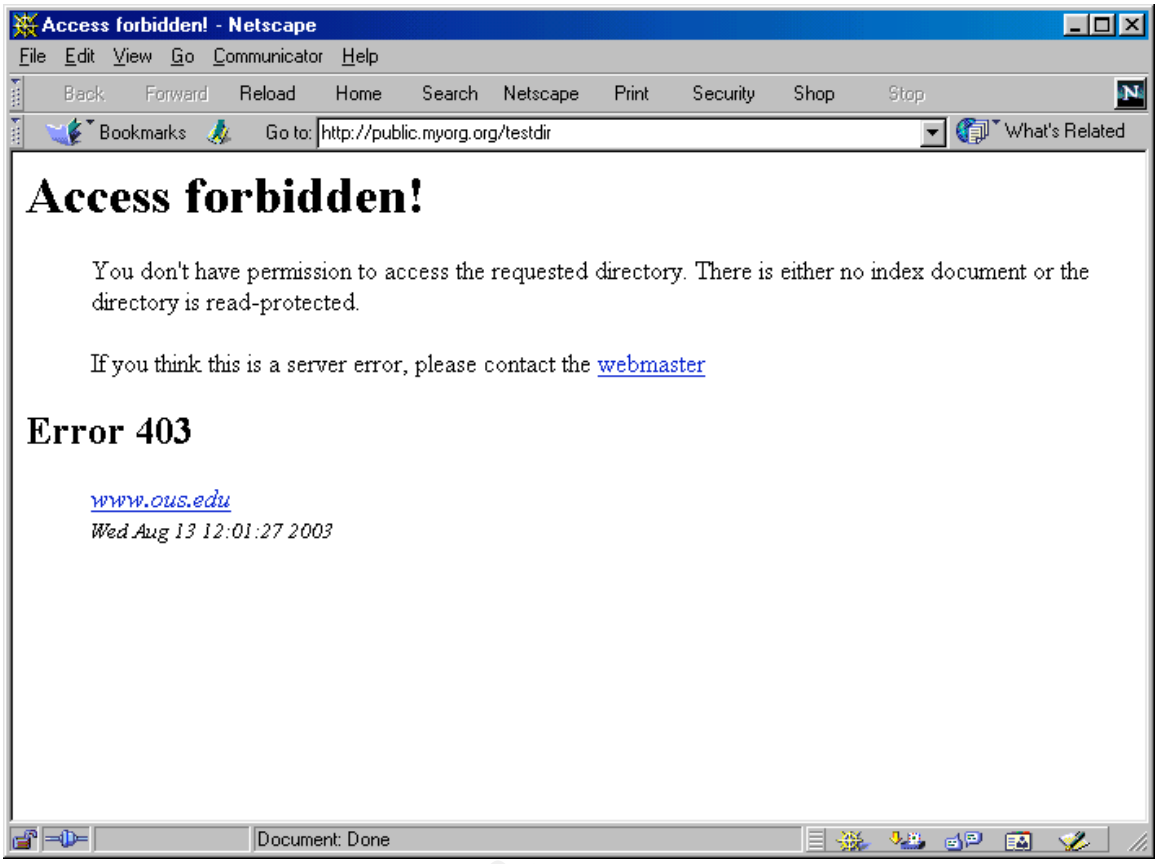
The process list appears to have everything we expect, and none that we don't. A clean process list is verified.

Sendmail Checks: We need to verify that Sendmail doesn't answer with more information than we want, and that we are not vulnerable to an open relay. To do this we connect to port 25, and make sure EXPN / VRFY don't respond. We also tried some combinations that would indicate that we are allowing relay through our system. All tests passed.

```
telnet public.myorg.org
Escape character is '^]'.
220 MyOrg ESMTP SMTP Gateway
11:47:02 -0700 (PDT)
helo outsideorg.org
250 public.myorg.org Hello hammerm@outsideorg.org [55.1.1.5], pleased
to meet you
VRFY hammersm
252 2.5.2 Cannot VRFY user; try RCPT to attempt delivery (or try finger)
EXPN hammersm
502 5.7.0 Sorry, we do not allow this operation
mail from:hammersm@myorg.org
250 2.1.0 hammersm@ myorg.org... Sender ok
rcpt to:hammerm@outsideorg.org
550 5.7.1 hammerm@outsideorg.org... Relaying denied
rcpt to:hammersm%hammersm@outsideorg.org
550 5.7.1 hammersm%hammersm@outsideorg.org... Relaying denied
rcpt to:hammersm%efn.org@outsideorg.edu
550 5.7.1 hammersm%efn.org@outsideorg.edu... Relaying denied
rcpt to:hammersm%efn.org@192.168.0.19
550 5.7.1 hammersm%efn.org@192.168.0.19... Relaying denied
quit
221 2.0.0 public.myorg.org closing connection
Connection closed by foreign host.
```

Apache checks: We want to make sure that apache doesn't list contents of a directory if no index exists. We also want to verify that if users tries to access the password change page without SSL it redirects them to the HTTPS version. We also want to verify that user cant set up a CGI to run from the web data content area.

First test we create a new folder in the webdata area called testdir and put some files in it. We attempt access it, and confirm that we do not get a directory listing:



We then attempt to access the password page via http, and confirm that it rewrote us to force connection to https.

Last we verify a user cant run a CGI in the document area. We put a cgi file in the document area and make it executable. When we attempt to access it the script is simply displayed in the browser.

Apache passes the initial tests. Testing for vulnerabilities as they come out, and monitoring of user content will be necessary to maintain security.

Bind: after connecting bind to the internet, verify we can't do a recursive lookup from the outside.

external% nslookup
Default Server: ns.external.org
Address: 66.1.1.26

> server public.myorg.org
Default Server: public.myorg.org
Address: 192.168.0.20

> private.myorg.org
Server: public.myorg.org

```
Address:  192.168.0.19

Name:    private.myorg.org
Address:  192.168.0.20

> www.msn.com
Server:  public.myorg.org
Address:  192.168.0.19
*** public.myorg.org can't find www.msn.com: Query refused
>exit
```

This verifies that

Sudo & SU: We want to verify that only users that are allowed to access Sudo and SU are able to. We test this with my "non admin" account. Since connections for this user via SSH are chrooted we use su to keep access to the whole system. We test the Chroot area next.

```
# su - hammerm
$ su
su: execute permission denied
$ /usr/local/bin/sudo cat /etc/shadow

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these two things:

    #1) Respect the privacy of others.
    #2) Think before you type.

Password:
hammerm is not in the sudoers file.  This incident will be reported.
```

We then check the mail to verify that the mail notice worked:

```
Subject: *** SECURITY information for public ***
Content-Length: 150
public : Aug 13 13:04:23 : hammerm : user NOT in sudoers ; TTY=pts/1 ;
PWD=/export/jail/home/hammerm ; USER=root ; COMMAND=/usr/bin/cat
/etc/shadow
```

Chroot / nosuid area: We want to connect via SSH as our test user and verify that we can't see outside of it. We also want to verify that SUID does not work on the /export volume. We create a suid copy of cat, and a root only file, shadows to test with.

```
cp /usr/bin/cat /export/jail/home/hammerm/cat
chmod 4755 /export/jail/home/hammerm/cat
```

```
touch /export/jail/etc/shadows
chown root:sys /export/jail/etc/shadows
chmod 700 /export/jail/etc/shadows
```

Now we ssh in as the test user and see what we can access.

```
$ ssh hammerm@public.myorg.org
Authorized uses only. All access may be logged.
hammerm@public.myorg.org's password:
Last login: Wed Aug 13 13:24:12 2003 from private.myorg.org
Sun Microsystems Inc.   SunOS 5.9       Generic May 2002
$ pwd
/home/hammerm
$ cat /etc/passwd
root:x:0:1:Super-User:/:/sbin/sh
hammerm:x:110:1:Mark User:/home/hammerm:/bin/sh
$ ls -l
total 28
-rwsr-xr-x   1 root     other      10284 Aug 13 13:39 cat
-rw-r--r--   1 hammerm  other        136 Aug 13 12:33 local.cshrc
-rw-r--r--   1 hammerm  other        157 Aug 13 12:33 local.login
-rw-r--r--   1 hammerm  other        174 Aug 13 12:33 local.profile
$ ls -l /etc
total 4
-rw-r--r--   1 root     other        356 Aug 13 12:44 group
-rw-r--r--   1 root     other         81 Aug 13 12:44 passwd
-rwx------   1 root     sys            0 Aug 13 13:39 shadows
$ ./cat /etc/shadows
cat: cannot open /etc/shadows
```

This verifies that as the jailed user it appears we are in a virtual root. The /etc folder has only the modified password file, which shows just the chroot /home/hammer home directory instead of the "real" one. When attempting to use a suid version of cat to access the test shadow file it shows that suid will not run.

Remote log checking & Slogger: Now we check to verify that remote logging works. Since the log server is reporting suspicious activity, we should have an alert from slogger about the failed sudo. Sure enough, we check and we have alerts from that and from our attempts in mail:

```
Security Violations
=-=-=-=-=-=-=-=-=-=-
Aug 13 11:47:30 public.myorg.org sendmail[6717]: [ID 801593 mail.info]
h7DIl20e006717:
hammerm@external.org [66.1.1.5]: VRFY hammersm [rejected]
Aug 13 11:47:33 public.myorg.org sendmail[6717]: [ID 801593 mail.info]
h7DIl20e006717:
```

hammerm@external.org [66.1.1.5]: EXPN hammersm [rejected]
Aug 13 14:27:49 public.myorg.org sudo: [ID 702911 local2.alert]
hammersm : user NOT in sudoers ; TTY=pts/1 ;
PWD=/export/jail/home/hammerm ; USER=root ; COMMAND=/usr/bin/cat
/etc/shadow

Attachments:

Appendix A:

Chroot patch: Save to /tmp/chroot-session.patch

```
###############################################################################
################
###############################################################################
################
#                                                             #
# Original patch by Ricardo Cerqueira <rmcc@clix.pt>                    #
#                                                             #
# Updated by James Dennis <james@firstaidmusic.com> for openssh-3.6.1
#
#                                                             #
# A patch to cause sshd to chroot when it encounters the magic token          #
# '/./' in a users home directory. The directory portion before the           #
# token is the directory to chroot() to, the portion after the                #
# token is the user's home directory relative to the new root.                #
#                                                             #
# Patch source using: patch -p0 < /path/to/patch                        #
#                                                             #
# Systems with a bad diff (doesn't understand -u or -N) should use gnu diff.   #
# Solaris may store this as gdiff under /opt/sfw/bin. I can't say much about   #
# other systems (unless you email me your experiences!).                      #
#                                                             #
###############################################################################
################
###############################################################################
################

diff -uNr openssh-3.6.1p1/session.c openssh-3.6.1p1-chroot/session.c
--- openssh-3.6.1p1/session.c   Thu Mar 20 20:18:09 2003
+++ openssh-3.6.1p1-chroot/session.c    Fri Apr  4 12:42:22 2003
@@ -58,6 +58,8 @@
 #include "session.h"
 #include "monitor_wrap.h"

+#define CHROOT
+
```

```
 #ifdef HAVE_CYGWIN
 #include <windows.h>
 #include <sys/cygwin.h>
@@ -1206,6 +1208,12 @@
 void
 do_setusercontext(struct passwd *pw)
 {
+
+#ifdef CHROOT
+       char *user_dir;
+       char *new_root;
+#endif /* CHROOT */
+
 #ifndef HAVE_CYGWIN
       if (getuid() == 0 || geteuid() == 0)
 #endif /* HAVE_CYGWIN */
@@ -1242,6 +1250,26 @@
                        exit(1);
                }
                endgrent();
+
+#ifdef CHROOT
+               user_dir = xstrdup(pw->pw_dir);
+               new_root = user_dir + 1;
+
+               while((new_root = strchr(new_root, '.')) != NULL) {
+                       new_root--;
+                       if(strncmp(new_root, "/./", 3) == 0) {
+                               *new_root = '\0';
+                               new_root += 2;
+
+                               if(chroot(user_dir) != 0)
+                                       fatal("Couldn't chroot to user directory %s", user_dir);
+                               pw->pw_dir = new_root;
+                               break;
+                       }
+                       new_root += 2;
+               }
+#endif /* CHROOT */
+
 # ifdef USE_PAM
                /*
                 * PAM credentials may take the form of supplementary groups.
```

Appendix B: pw1.pl script

```
#!/usr/local/bin/perl

# fix path to remove taint checks
$ENV{PATH} = '/usr/local/bin:/bin:/usr/bin';
$ENV{IFS} = '';

# Prevent buffering problems
$| = 1;

use CGI qw/:standard :html3/;
use Socket;

# display the HTML header
print header,
      start_html(-title=>'Change www Password',
            -bgcolor=>'white');

import_names('Q');

TRY: {
   last TRY unless $Q::user;
   my ($rv,$msg) = check_consistency();
   do_error($msg),last TRY unless $rv;

   ($rv,$msg) = set_passwd($Q::user,$Q::old,$Q::new1);
   do_error($msg),last TRY unless $rv;

   print $msg;
   $OK++;
}

create_form() unless $OK;

print
      p,
      a({href=>'/'},'Home page'),
      hr,
      a('Contact Webadmin for assistance'),
      end_html;

sub check_consistency {
  return (undef,'Please fill in the user name field.') unless $Q::user;
# only allow changing of user accounts (101 to 9999)
  my $uid1 = (getpwnam($Q::user))[2];
  return (undef,"Illegal username of $uid1") unless ($uid1>100 && $uid1<9999);
  return (undef,'Please fill in the old password field.') unless $Q::old;
```

```
  return (undef,'Please fill in the new password fields.') unless $Q::new1 &&
$Q::new2;
  return (undef,"New password fields don't match.") unless $Q::new1 eq
$Q::new2;
# only allow valid lengths
  return (undef,"Suspicious user name $Q::user.") unless $Q::user=~/^\w{3,8}$/;
  return (undef,'Invalid old password.') unless ((length($Q::old) < 17) &&
(length($Q::old) > 5));
  return (undef,'Invalid new password.') unless ((length($Q::new1) < 17) &&
(length($Q::new1) > 5));
}

sub set_passwd ($$$) {
my($uname,$oldpw,$newpw) = @_;

my $socket_file = '/var/wtmp/pwchanged_sock';

socket(SOCK, PF_UNIX, SOCK_STREAM, 0) or die return (undef,"Password
failed. Socket open failed.  Contact webadmin");
connect(SOCK, sockaddr_un($socket_file)) or die return (undef,"Password failed.
Socket connect failed.  Contact webadmin");

send SOCK, "Username: $uname; Old Password: $oldpw; New Password:
$newpw\n", 0;

while (my $line = <SOCK>)
{
    chomp $line;
    print "$line<BR>\n";
}

print "</body>\n</html>\n\n";

return (1,$line);
}

sub create_form {
#print
print <<EOF;
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US">
<head><title>Change Web password form</title> </head>
<body bgcolor="#21FFEE"><h1><a href="/"><img align="left" width="80"
height="64" alt="Home" border="0" src="../images/LogoLsmall.gif" /></a> Web
Access: Change web server password <br clear="all" /></h1><p>Please fill out
all fields before submitting.</p><form method="post" action="pw1.pl"
enctype="application/x-www-form-urlencoded">
```

```
<hr /><table><tr><th align="right">www username:</th> <td><input type="text"
name="user"  size="10" maxlength="8"/></td> <td> </td></tr> <tr><th
align="right">Current password:</th> <td><input type="password" name="old"
size="18" maxlength="16" /></td> <td> </td></tr> <tr><th
colspan="3"> </th></tr> <tr><th align="right">New password:</th>
<td><input type="password" name="new1"  size="18" maxlength="16" /></td>
 <td rowspan="2">Your password serves as a security check to protect your
account from unauthorized users.  <em><strong>Do not share your
password!</strong></em>.  Your password should be at least 6 characters, with
a mix of letters and non letters.  Try to make a difficult to guess password,
possibly using mixed capitalization an symbols, e.g. gUe\$\$mE\!
</td></tr> <tr><th align="right">Confirm new password:</th> <td><input
type="password" name="new2"  size="18" maxlength="16"/></td></tr>
<tr><td> </td> <th align="left"><input type="submit" name="action"
value="Change password" /></th> <td><input type="Reset" name=".defaults"
value="Clear form" /></td></tr></table>
<hr>
EOF
}

sub do_error ($) {
   print "<CENTER><FONT COLOR='red' SIZE='+1'>";
   print b('Error: '),shift," Password is not changed.";
   print "</FONT></CENTER>";
}
```

Appendix B: pwd.pl script

```
#!/usr/local/bin/perl

# fix path to remove taint checks
$ENV{PATH} = '/usr/local/bin:/bin:/usr/bin';
$ENV{IFS} = '';

# Prevent buffering problems
$| = 1;

# pwchanged
# Password Change Daemon
# This daemon should run as root and accept socket connections
# from CGI scripts, allowing users to change their passwords
# via the Web.  Adopted from a partial script written by Kai MacTane
# Changing: Version 2.0 will run as a non-priveleged user and use
# expect to change passwords.

use strict;
```

```
use Socket;
use Unix::Syslog qw(:macros);  # Syslog macros
use Unix::Syslog qw(:subs);    # Syslog functions
use CGI qw/:standard :html3/;

# set pwgroup and www id, used to restrict who can write to
# the perl socket to just these.  Setting hard numbers here rather
# than reading them in to avoid clobbering.
my $pwgroup = 14;
my $wwwid = 108;

my ($syslog_facility, $syslog_options, $syslog_priority);
my ($username, $old_pass, $new_pass);

# What syslog facility should we use?
# Using 'authpriv' keeps messages from being logged anywhere
# except /var/log/secure.
$syslog_facility = '1<<3';   # equals 'user' facility.

# What syslog priority should we use?
$syslog_priority = '6';      # equals 'info' level.

# Any syslog options?
$syslog_options = 'LOG_PID';

# Socket file going in a secure driectory, only user www can access
my $socket_file = '/var/wtmp/pwchanged_sock';

# And how many connections should we queue?
my $max_conn = 2;

# End of variables.

&write_syslog('Starting changepw daemon.');

# Check that socket file exists and is world-writable.
### (Yet to be written).

# Now fork the daemon process into the background.
unless (fork)
  {
  unless (fork)
    {
    # Daemon process should now be running in background,
#    &write_syslog('pwchanged Daemon process now forked.');
# BIGLOOP:
```

```
   while (1)
   {
     # Now you need to wait for sockets to be formed
     # by the CGI script client.

     my $uaddr = sockaddr_un($socket_file);
     my $proto = getprotobyname('tcp');

     socket(Sock, PF_UNIX, SOCK_STREAM, 0)    or die "No socket: $!";
     unlink($socket_file);
     bind(Sock, $uaddr);
     listen(Sock, $max_conn);
     # Restricting socket file permissions
     chown $wwwid, $pwgroup, $socket_file;
     chmod 0770, $socket_file;

     &write_syslog('Daemon listening on %s for <= %d connections.', $socket_file,
$max_conn);

     # (Then some kind of parsing on incoming msgs...)

     &write_syslog('Entering main FOR loop.');
LOOP:
# waiting for input
     for ( ; my $idler = accept(CLIENT, Sock); close CLIENT)
     {

       my $line = <CLIENT>;
       chomp $line;

       &write_syslog('Client connection');

       ($username, $old_pass, $new_pass) = split(/\;\s*/, $line);
       if ($username =~ /^Username:\s*/)
       {
         $username = $';
       } else {
         print CLIENT "Invalid calling syntax!\n";
         &write_syslog('Invalid calling syntax from client; dropping connection.');
         next LOOP;
       }

     if ($old_pass =~ /^Old Password:\s*/)
     {
       $old_pass = $';
     } else {
```

```
      print CLIENT "Invalid calling syntax!\n";
      &write_syslog('Invalid calling syntax from client; dropping connection.');
      next LOOP;
    }

    if ($new_pass =~ /^New Password:\s*/)
    {
      $new_pass = $';
    } else {
      print CLIENT "Invalid calling syntax!\n";
      &write_syslog('Invalid calling syntax from client; dropping connection.');
      next LOOP;
    }

# Now do some checks on the incoming data in case someone is trying to
# hack.  These should have mostly been caught by the front-end perl app,
# so if they trap an error that should have been caught, the demon aborts
# and sends a warning to syslog.

    if ((length($username)>8) or (length($new_pass)>16) or
(length($old_pass)>16) or (length($username)<4) or (length($new_pass)<6) or
(length($old_pass)<6))
    {
      print CLIENT "Invalid input: aborted. \n";
      &write_syslog('ALERT: Illegal PWChange input length, aborting!');
      exit (0);
    }

    # We should now have a tested $username, $old_pass, $new_pass.

    # Make sure the user exists.
    unless (`grep ^$username\: /etc/passwd`)
    {
      print CLIENT "No such user: $username. Aborting.\n";
#       &write_syslog('Invalid user "%s"; dropping connection.', $username);
      next LOOP;
    }

    if ($new_pass eq $old_pass)
    {
       print CLIENT "New password must be different from old password!\n";
#         &write_syslog('Error: old and new passwords matched for user %s',
$username);
       next LOOP;
    }
```

```
# Now get the UID, abort if invalid:
     my ($user1,$passwd1,$uid1,$junk);
     ($user1,$passwd1,$uid1,$junk)=getpwnam($username);
     if (($uid1 < 100) or ($uid1 > 9999))
     {
     print CLIENT "Attempting to change illegal user from WEB \n";
#      &write_syslog('PWChanged attempting to change illegal user';
#      exit(0);
     }

# Everything looks ok, now try to change pasword
     my ($rv,$msg)=set_passwd($username,$old_pass,$new_pass);

     if ($rv==1) {
       print CLIENT ("Password changed: $msg"); }
     else {
       print CLIENT ("Password change failed: <STRONG>$msg
</STRONG><br><a href=\"/cgi-bin/pw1.pl\">Change Password</a>");
     }

     # Make sure we don't get in a tight loop.
     sleep 5;
   }
          &write_syslog('Exited main FOR loop.');
 }

 &write_syslog('Exiting at point 1.');
 exit 0;
}
#&write_syslog('Exiting at point 2.');
exit 0;
}

wait;

sub write_syslog
{
    my $format = shift;
    openlog('pwchanged', $syslog_options, $syslog_facility);
    syslog($syslog_priority, $format, @_);
    closelog();

}

sub set_passwd ($$$) {
   use Expect;
```

```
    $Expect::Log_Stdout=0;
# to enable logging:
#    $Expect::Log_Stdout=1;
#    $Expect::Debug=3;
#    $Expect::Exp_Internal=1;

    my $TIMEOUT = 5;
    my $PASSWD = "/bin/passwd";
    my $SU = '/bin/su';

    my($user,$old,$new) = @_;

    # spawn the su command using expect
    my $passwd = Expect->spawn("$SU $user -c \"$PASSWD $user\"") || return
(undef,"Couldn't open $SU $user -c $PASSWD");
    # wait for su to prompt for password
    my $rv = $passwd->expect($TIMEOUT,
                    "Password:",
                    "su: Unknown id: $user");
    $rv == 2 && return (undef,"User $user unknown.");
    $rv || return (undef,"Didn't get su password prompt.");

    # send the password for the user
    print $passwd "$old\r";

    # wait for passwd to prompt for old password
    $rv = $passwd->expect($TIMEOUT,
                    "Enter login password:",
                    "su: Sorry");
    $rv == 2 && return (undef,"Old password is incorrect.");
    $rv || return (undef,"Didn't get prompt for old password.");

    # print old password
    print $passwd "$old\r";
    $rv = $passwd->expect($TIMEOUT,
                    "New password:",
                    "Sorry.");
    $rv == 2 && return (undef,"Old password is incorrect.");
    $rv || return (undef,"Timed out without seeing prompt for new password.");

    # print new password
    print $passwd "$new\r";
    $rv = $passwd->expect($TIMEOUT,
                    "Re-enter new password:",
                    "Please use a longer password.",
```

```
                "Passwords must differ by at least 3 positions",
                "Password must contain at least two alphabetic characters and at
least one numeric or special character.");
   $rv == 2 && return (undef,"Please use a longer password.");
   $rv == 3 && return (undef,"Passwords must differ by at least 3 positions.");
   $rv == 4 && return (undef,"Password must contain at least two alphabetic
characters and at least one numeric or special character.");
   $rv || return (undef,"Timed out without seeing second prompt for new
password.");

   # reconfirm password
   print $passwd "$new\r";
   $rv = $passwd->expect($TIMEOUT,
                "They don't match; try again.");
   $rv == 1 && return (undef,"Second password doesn't match.");

   $passwd->hard_close();

   return (1,"Password changed successfully for $user.");
}
```

Appendix D: chkproc script

```
#!/bin/sh
#
# Monitor processes - restart if necessary
#
# syslogd
#
pidfile=/etc/syslog.pid
procname=syslogd
procinit="/etc/init.d/syslog start"
#
#
if [ -f $pidfile ]; then
        procpid=`cat $pidfile`
#
        if [ "$procpid" -gt 0 ]; then
                chkproc=`/usr/bin/ps -p $procpid 2>&1 | /usr/bin/grep -v "TTY" |
/usr/bin/awk '{print $1}'`
                if [ "x${chkproc}" = "x" ]; then
                        echo "$procname process has died - restarting\n"
                        $procinit
#                        /etc/init.d/$procinit start
                fi
        fi
```

```
else
        echo "No $pidfile found, assuming $procname process has died -
restarting\n";
        $procinit
fi
#
# named
#
pidfile=/etc/named.pid
procname=named
procinit="/usr/local/sbin/named restart"
#
#
if [ -f $pidfile ]; then
        procpid=`cat $pidfile`
#
        if [ "$procpid" -gt 0 ]; then
                chkproc=`/usr/bin/ps -p $procpid 2>&1 | /usr/bin/grep -v "TTY" |
/usr/bin/awk '{print $1}'`
                if [ "x${chkproc}" = "x" ]; then
                        echo "$procname process has died - restarting\n"
                        $procinit
                fi
        fi
else
        echo "No $pidfile found, assuming $procname process has died -
restarting\n";
        $procinit
fi
#
# apache
#
pidfile=/usr/local/apache2/logs/httpd.pid
procname=httpd
procinit="/usr/local/apache2/bin/apachectl startssl"
#
#
if [ -f $pidfile ]; then
        procpid=`cat $pidfile`
#
        if [ "$procpid" -gt 0 ]; then
                chkproc=`/usr/bin/ps -p $procpid 2>&1 | /usr/bin/grep -v "TTY" |
/usr/bin/awk '{print $1}'`
                if [ "x${chkproc}" = "x" ]; then
                        echo "$procname process has died - restarting\n"
                        $procinit
```

```
                fi
        fi
else
        echo "No $pidfile found, assuming $procname process has died -
restarting\n";
        $procinit
fi
#
# sendmail
#
pidfile=/var/run/sendmail.pid
procname=sendmail
procinit="/etc/init.d/sendmail start"
#
#
if [ -f $pidfile ]; then
        procpid=`cat $pidfile`
#
        if [ "$procpid" -gt 0 ]; then
                chkproc=`/usr/bin/ps -p $procpid 2>&1 | /usr/bin/grep -v "TTY" |
/usr/bin/awk '{print $1}'`
                if [ "x${chkproc}" = "x" ]; then
                        echo "$procname process has died - restarting\n"
                        $procinit
                fi
        fi
else
        echo "No $pidfile found, assuming $procname process has died -
restarting\n";
        $procinit
fi
```

Appendix D: rosyslog script:

```
#!/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

while [ $DEPTH -gt 0 ]
do
    OLD=$(($DEPTH -1 ))
    if [ -f $LOG.$OLD ]; then
        mv $LOG.$OLD $LOG.$DEPTH
    fi
    DEPTH=$OLD
done

if [ $DEPTH -eq 0 -a -f $LOG ]; then
    mv $LOG $LOG.0
fi

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

/etc/rc2.d/S74syslog stop
/etc/rc2.d/S74syslog start
```

Appendix F: roaplogs script:

```
#!/usr/bin/ksh
# roaplogs : script to rotate apache logs to a folder with the current date as the
name
# and then zip and migrate those logs to an archive area

FIND="/usr/bin/find"

LOGPATH="/usr/local/apache2/logs"
ARCHPATH="/export/jail/aplogs"
STOPCMD="/usr/local/apache2/bin/apachectl stop"
STARTCMD="/usr/local/apache2/bin/apachectl startssl"

#stop the apache server so we can move the logs
$STOPCMD

SUBDIR=`date | awk '{print $6 $2 $3}'`

if [ ! -d $LOGPATH ]; then
```

```
  echo "$DIR: Path does not exist"
  exit 255
fi

cd $LOGPATH

if [ -d $SUBDIR ]; then
  echo "$SUBDIR: Backup Log Dir Exists, Aborting"
  exit 255
fi

#first we move the files into a subdirectory of logs
#to minimize downtime of the server
mkdir $SUBDIR

for file in `$FIND . -type f`
do
  mv $file $SUBDIR/$file
done

#now that the logs are moved, start the server
$STARTCMD

#now that apache is back up, zip the logs and migrate
#them to the archive path on the users volume
cd $SUBDIR
for file in `$FIND . -type f`
do
  zip $file.zip $file
  rm $file
done
cd $LOGPATH
mv $SUBDIR $ARCHPATH
```

Appendix G: Nessus results

```
Nessus Scan Report
------------------
SUMMARY

- Number of hosts which were alive during the test : 1
- Number of security holes found : 0
- Number of security warnings found : 7
- Number of security notes found : 23

TESTED HOSTS
```

192.168.0.19 (Security warnings found)

DETAILS

+ 192.168.0.19 :
. List of open ports :
  o ssh (22/tcp) (Security notes found)
  o smtp (25/tcp) (Security notes found)
  o domain (53/tcp) (Security warnings found)
  o http (80/tcp) (Security warnings found)
  o https (443/tcp) (Security warnings found)
  o ntp (123/udp) (Security warnings found)
  o domain (53/udp) (Security notes found)
  o general/udp (Security notes found)
  o general/tcp (Security warnings found)
  o general/icmp (Security warnings found)

. Information found on port ssh (22/tcp)

  An ssh server is running on this port

. Information found on port ssh (22/tcp)

  The remote SSH daemon supports the following versions of the
  SSH protocol :

    . 1.99
    . 2.0

. Information found on port ssh (22/tcp)

  Remote SSH version : SSH-2.0-OpenSSH_3.6.1p2

. Information found on port smtp (25/tcp)

  An SMTP server is running on this port
  Here is its banner :
  220 MyOrg ESMTP SMTP Gateway

. Information found on port smtp (25/tcp)

  Remote SMTP server banner :
  220 MyOrg ESMTP SMTP Gateway

. Information found on port smtp (25/tcp)

smtpscan was not able to reliably identify this server. It might be:
Sendmail 8.12.8/8.11.6 -14-
Sendmail 8.11.6/8.11.6 -474-
Sendmail 8.11.6/8.11.6 -309-
Sendmail 8.11.6
Sendmail Switch-2.2.3
Sendmail 8.11.0/8.11.0
Sendmail 8.12.9/8.12.9 -152-
The fingerprint differs from these known signatures on 2 point(s)

If you known precisely what it is, please send this fingerprint
to the Nessus team :
:503:501:501:250:553:553:550:502:252:502:502:502:502:250:250

. Information found on port smtp (25/tcp)

Nessus sent several emails containing the EICAR
test strings in them to the postmaster of
the remote SMTP server.

The EICAR test string is a fake virus which
triggers anti-viruses, in order to make sure
they run.

Nessus attempted to e-mail this string five times,
with different codings each time, in order to attempt
to fool the remote anti-virus (if any).

If there is an antivirus filter, these messages should
all be blocked.

*** To determine if the remote host is vulnerable, see
*** if any mail arrived to the postmaster of this host

Solution: Install an antivirus / upgrade it

Reference : http://online.securityfocus.com/archive/1/256619
Reference : http://online.securityfocus.com/archive/1/44301
Reference : http://online.securityfocus.com/links/188

Risk factor : Low

. Warning found on port domain (53/tcp)

The remote name server allows recursive queries to be performed

by the host running nessusd.

If this is your internal nameserver, then forget this warning.

If you are probing a remote nameserver, then it allows anyone
to use it to resolve third parties names (such as www.nessus.org).
This allows hackers to do cache poisoning attacks against this
nameserver.

See also : http://www.cert.org/advisories/CA-1997-22.html

Solution : Restrict recursive queries to the hosts that should
use this nameserver (such as those of the LAN connected to it).
If you are using bind 8, you can do this by using the instruction
'allow-recursion' in the 'options' section of your named.conf

If you are using another name server, consult its documentation.

Risk factor : Serious
CVE : CVE-1999-0024
BID : 678

. Information found on port domain (53/tcp)

A DNS server is running on this port. If you
do not use it, disable it.

Risk factor : Low

. Information found on port domain (53/tcp)

The remote bind version is : 9.2.2

. Warning found on port http (80/tcp)

Your webserver supports the TRACE and/or TRACK methods. It has been
shown that servers supporting this method are subject
to cross-site-scripting attacks, dubbed XST for
'Cross-Site-Tracing', when used in conjunction with
various weaknesses in browsers.

An attacker may use this flaw to trick your
legitimate web users to give him their
credentials.

Solution: Disable these methods.

If you are using Apache, add the following lines for each virtual
host in your configuration file :

```
RewriteEngine on
RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK)
RewriteRule .* - [F]
```

If you are using Microsoft IIS, use the URLScan tool to deny HTTP TRACE
requests or to permit only the methods needed to meet site requirements
and policy.

See http://www.whitehatsec.com/press_releases/WH-PR-20030120.pdf
http://archives.neohapsis.com/archives/vulnwatch/2003-q1/0035.html

Risk factor : Medium

. Information found on port http (80/tcp)

A web server is running on this port

. Information found on port http (80/tcp)

The following directories were discovered:
/cgi-bin, /icons, /manual, /test

. Information found on port http (80/tcp)

The remote web server type is :

Apache

and the 'ServerTokens' directive is ProductOnly
Apache does not permit to hide the server type.

. Warning found on port https (443/tcp)

Your webserver supports the TRACE and/or TRACK methods. It has been
shown that servers supporting this method are subject
to cross-site-scripting attacks, dubbed XST for
'Cross-Site-Tracing', when used in conjunction with
various weaknesses in browsers.

An attacker may use this flaw to trick your
legitimate web users to give him their
credentials.

Solution: Disable these methods.

If you are using Apache, add the following lines for each virtual host in your configuration file :

```
RewriteEngine on
RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK)
RewriteRule .* - [F]
```

If you are using Microsoft IIS, use the URLScan tool to deny HTTP TRACE requests or to permit only the methods needed to meet site requirements and policy.

See http://www.whitehatsec.com/press_releases/WH-PR-20030120.pdf
http://archives.neohapsis.com/archives/vulnwatch/2003-q1/0035.html

Risk factor : Medium

. Warning found on port https (443/tcp)

The SSLv2 server offers 6 strong ciphers, but also
0 medium strength and 2 weak "export class" ciphers.
The weak/medium ciphers may be chosen by an export-grade
or badly configured client software. They only offer a
limited protection against a brute force attack

Solution: disable those ciphers and upgrade your client
software if necessary

. Information found on port https (443/tcp)

A TLSv1 server answered on this port

. Information found on port https (443/tcp)

A web server is running on this port through SSL

. Information found on port https (443/tcp)

The following directories were discovered:
/cgi-bin, /icons, /manual, /test

. Information found on port https (443/tcp)

The remote web server type is :

Apache

and the 'ServerTokens' directive is ProductOnly
Apache does not permit to hide the server type.

. Information found on port https (443/tcp)

Here is the SSLv2 server certificate:
Certificate:
Data:
Version: 1 (0x0)
Serial Number: 0 (0x0)
Signature Algorithm: md5WithRSAEncryption
Issuer: C=US, ST=oregon, L=corvallis, O=myorg, OU=myorg,
CN=public.myorg.org/emailAddress=webadmin@myorg.org
Validity
Not Before: Jul 31 18:40:13 2003 GMT
Not After : Apr 25 18:40:13 2006 GMT
Subject: C=US, ST=oregon, L=corvallis, O=myorg, OU=myorg,
CN=public.myorg.org/emailAddress=webadmin@myorg.org
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)
Modulus (1024 bit):
00:ce:57:ec:34:c5:06:a2:91:66:8a:85:76:75:27:
9d:85:be:fd:3e:36:3a:8e:92:22:2e:73:15:69:64:
1f:ec:2e:bb:a4:e2:a2:d8:b3:74:54:a8:2a:1f:ed:
25:5e:09:d0:34:6f:86:1d:4f:a3:6e:1a:03:3b:62:
88:50:06:f6:1b:61:5e:33:1e:79:ff:03:f9:99:59:
38:11:0a:7c:2e:72:b7:06:05
Exponent: 65537 (0x10001)
Signature Algorithm: md5WithRSAEncryption
37:3f:fd:c1:68:7c:67:0c:51:8f:03:9a:27:1d:91:9e:80:cf:
e3:cd:85:c5:5b:ab:15:9c:64:95:3d:5f:7e:f4:9d:1d:06:e1:
64:32:7a:78:e2:74:5a:78:f8:23:81:57:cd:a8:c1:95:24:87:
f2:3e:61:78:35:27:26:91:2f:b2:92:da:60:80:91:d6:28:73:
d3:bf:4a:c1:67:3b:26:5b:9e:86:ca:db:be:39:0e:5f:99:21:
f4:ae

. Information found on port https (443/tcp)

Here is the list of available SSLv2 ciphers:
RC4-MD5
EXP-RC4-MD5

RC2-CBC-MD5
EXP-RC2-CBC-MD5
IDEA-CBC-MD5
DES-CBC-MD5
DES-CBC3-MD5
RC4-64-MD5

. Information found on port https (443/tcp)

This TLSv1 server also accepts SSLv2 connections.
This TLSv1 server also accepts SSLv3 connections.

. Warning found on port ntp (123/udp)

An NTP server is running on the remote host. Make sure that
you are running the latest version of your NTP server,
has some versions have been found out to be vulnerable to
buffer overflows.

*** Nessus reports this vulnerability using only
*** information that was gathered. Use caution
*** when testing without safe checks enabled.

If you happen to be vulnerable : upgrade
Solution : Upgrade
Risk factor : High
CVE : CVE-2001-0414
BID : 2540

. Information found on port ntp (123/udp)

It is possible to determine a lot of information about the remote host
by querying the NTP variables - these include OS descriptor, and
time settings.

Theoretically one could work out the NTP peer relationships and track back
network settings from this.

Quickfix: Set NTP to restrict default access to ignore all info packets:
 restrict default ignore

Risk factor : Low

. Information found on port domain (53/udp)

A DNS server is running on this port. If you

do not use it, disable it.

Risk factor : Low

. Information found on port general/udp

For your information, here is the traceroute to 192.168.0.19 :
192.168.0.19

. Warning found on port general/tcp

The remote host does not discard TCP SYN packets which
have the FIN flag set.

Depending on the kind of firewall you are using, an
attacker may use this flaw to bypass its rules.

See also : http://archives.neohapsis.com/archives/bugtraq/2002-10/0266.html
http://www.kb.cert.org/vuls/id/464113

Solution : Contact your vendor for a patch
Risk factor : Medium
BID : 7487

. Information found on port general/tcp

Remote OS guess : Solaris 9 with TCP_STRONG_ISS set to 2

CVE : CAN-1999-0454

. Warning found on port general/icmp

The remote host answered to an ICMP_MASKREQ
query and sent us its netmask (255.255.255.0)

An attacker can use this information to
understand how your network is set up
and how the routing is done. This may
help him to bypass your filters.

Solution : reconfigure the remote host so
that it does not answer to those requests.
Set up filters that deny ICMP packets of
type 17.

Risk factor : Low

CVE : CAN-1999-0524

This file was generated by the Nessus Security Scanner

Appendix H: References:

Pomeranz, Hal Solaris Security Step by Step (Version 2.0)
The Sans Institute, 2001

Watson, Keith A. nddconfig (script containing descriptions for nddconfig)
Sun Microsystems, Inc. 1999
URL: http://www.fish.com/titan/arch/sol2sun4/lib/nddconfig

Noordergraaf, Alex & Watson, Keith
Solaris Operating Environment Security - Updated for Solars 8 OS
Sun Blueprints Online - April 2001
URL: http://www.sun.com/solutions/blueprints/0401/security-updt1.pdf

Noordergraaf, Alex & Osser, William
Auditing in the Solaris 8 Operating Environment
Sun Blueprints Online - February 2001
URL: http://www.sun.com/solutions/blueprints/0201/audit_config.pdf

Cowgill, Kent SSHD Chrooted install / setup
(kent@c2group.net), URL: http://www.c2group.net/howtos/chroot.html

MacTane, Kai (kmactane@GothPunk.com)
Change Password discussions / newsgroup thread
from grrltalk-techtalk 2001
URL: http://www.ultraviolet.org/archive/grrltalk-techtalk.2001/

Sans Reading Room:
SANS Institute 2003
URL: http://www.sans.org/rr/