Global Information Assurance Certification Paper

SANS GIAC Certification

Building a Secure Solaris 8 Backup Server

GCUX Practical Assignment (v.1.8) (October 2001)

Jason Christensen

Introduction:

Backups are a necessary task in any computing environment. The vulnerability exposures in setting up a backup server are largely decreased if proper hardening steps are taken. Beyond the normal benefits of a backup server which include: recovering hosts from failed hardware, administrator mistakes, or those not so smooth software upgrades, backups can provide an audit trail of a client host in the event of a security incident. You will be able chronologically archive and retrieve filesystem/application snapshots that can aid in determining exactly when a problem/incident occurred. Performing this task securely, and with simple secure tools is the goal of this document.

Several assumptions about doing secure backups are used:

1. Backups are frequent.
2. Restores are infrequent.
3. Backups need to be secure and are automated.
4. Restores need to be secure and are done by a trusted administrator.

This document has the assumptions that you have a basic knowledge of UNIX concepts and are familiar with the vi editor. This document also assumes that you have access to a known trusted solaris 8 server where you can compile and configure software and write it to CD for purposes of installation on your secure backup server. You should also identify all your network services that you will be using including DNS servers and network timeservers. Obtaining permission from your network administrator and heeding recommendations on which DNS servers and network timeservers you should use is also assumed.

The security risks of this server include comprising the backup server directly and/or comprising the backup data as it is transmitted over the network. Attacking this backup server directly include remote attacks and unauthorized physical access to this host. Using standard network encryption techniques mitigates compromising backup data as it is being transmitted over the network.

Encryption is at the core of this security document. Do not assume that a switched network provides sufficient network data protection. Assume that it is possible for someone who is determined, to view and log all packets coming to or from this host and make security decisions based on that paranoid view. Administrator vigilance is crucial in setting up and securely maintaining a backup server.

System Configuration:

Hardware:

The server for this backup practical is a SUN Enterprise-250 running Solaris 8 with 2-400 Mhz processors, 1 gig of memory, 2 power supplies, internal cdrom, internal 4mm DDS3 DAT drive, and an external scsi-attached 30 tape DLT-7000 tape library (Storagetek Timberwolf 9730). The hard disks in this host are 1 9-gig drive and 1 74-gig drive.

Physical Location:

The server and tape library may contain backups of many of your most secure DMZ firewalls and important application hosts. Do not make this the weak link in the chain. Comprising this server would compromise all data on these secure hosts. Worse, a compromised filesystem restore could propagate a security break-in across your enterprise. Therefore, several security/hardening precautions will be taken.

Physical security for this server is the first thing that should be verified. A locked room with limited access and a lock on the server and tape library are a minimum. Also do not overlook the importance of properly storing backup tapes when they are removed from the tape library for off-site storage. Accounting for these tapes is the backup administrators responsibility. You want to be able to get access to these tapes in a relatively short time, but you want to make sure that proper authorization steps are taken for anyone who requests these tapes. These steps should include written agreements with specific details about which personnel and when tapes can be retrieved from off-site storage along with dates when tape return is expected. I suggest finding an offsite storage vendor that allows you to access your tape storage inventory online via a secure interface so you can periodically compare what should/shouldn't be at the off-site location. You should also get an e-mail or similar notification when tapes are moved to or from off-site storage.

The server and tape library for this practical are installed and placed in a secure underground data center. The server is on redundant building PDU power and also has as its own battery UPS in the event of PDU power failure. Biometric scanners, mantraps, and cameras are used to secure the server from unauthorized physical access. The cameras are positioned so they clearly show the server and tape library as they record any physical access to the room. This host's keyboard and monitor are carefully placed so that the security cameras do not view the screen or record keyboard input. If your secure room has recorded card key access it is advised to get a monthly report of the accesses and review it for anomalies and failed access attempts. Also if possible, pick a secure room with low personnel traffic as it makes auditing easier.

Network Location:

This server is going to be placed on a secure internal network. This paper will not cover specific network layouts or firewall products/concepts but keep in mind several design issues as it is likely that at least one firewall will be in your network path between your server and its present or future clients:

A connection/rule either TO or FROM (but not necessarily in both directions) the backup server and its clients will need to be allowed if a firewall is in the network path.

a. To the backup server if backup connection is client initiated.
b. From the backup server if backup connection is server initiated.

If the server or this network segment is protected by its own firewall as it very well might be in a secure environment it should block all other hosts from making connections to or from this backup server even at the expense of having to change the rule or rules sets each time a new backup client is added. An example of a client initiated backup will be shown later in this document via ssh (secure shell) and in that example only port 22 (default for ssh) from client to server is needed to accomplish this backup.

For the purposes of this document there is no firewall and steps are taken to hardened this server against remote vulnerability attacks. This includes kernel network settings, removing unneeded network services, and the implementing access lists for services we are running.

Performing backups can be very network intensive and placing the server on a busy network segment may slow backups, adjacent servers, and other services on that network segment.

Installation Steps:

1. Start Secure:

Perform the installation in a secure physical area.
Unplug the server from any network connections.
Power-on or reboot the server.

2. Open Boot Settings:

On boot up press Stop-A to get to the “ok” prompt

Make the following changes to the Open Boot Prom environment.

a. ok **setenv oem-banner "Authorized Access Only. All access may be logged"**
(This is a warning message for users that will help identify and prosecute misuse of this server)
b. ok **setenv oem-banner? True**
(Display banner)
c. ok **passwd** (system will prompt you for an OBP password. Share this password with other trusted administrators)
d. ok **setenv security-mode full**
(This will turn on the security-mode features in OBP. The host will now require a password for OBP commands and is an additional precaution against unauthorized physical access changes. This password is written to the OBP NVRAM memory and a physical chip will need to be ordered and replaced if the password is forgotten.)

3. Initial Install/Re-install of this host:

.
Install the Solaris Operating System from trusted media

a. Insert the **Solaris 8 Software Disk 1** of 2 cdrom in to the cd drive.
b. At the "ok" prompt, type boot cdrom
ok **boot cdrom**

4. Installation OS/Network Settings:

When prompted for the following enter *your* appropriate values:

a.	Language	(English)
b.	Locale	(English C – 7 Bit ASCII)
c.	Networked	(Yes)
d.	Use DHCP	(No)
e.	Host name	(sansbackup)
f.	IP address	(10.10.10.1)
g.	System part of a subnet	(Yes)
h.	Netmask	(255.255.255.0)
i.	Enable IPv6	(No)
j.	Configure Kerberos Security	(No)
k.	Name service	(None)
l.	Specify Time zone	(Geographic region US/Central)
m.	Date and time	(Correct data or time if incorrect)
n.	Initial/Upgrade Solaris OS	(Be sure to select: Initial)

Don't select any additional Geographic regions (Continue)
Select 64-bit operating system support (It should already be checked)
Select "Core System Support" as your Software Group to install.

5. Disk Layouts and Customization:

Select your boot disk (c0t0d0)
Do not choose to preserve any old filesystems and do not choose to mount any remote filesystems.

Automatically Layout File Systems (Select "Auto Layout")
Choose to create filesystems: /, /usr, /var, & swap.

Filesystem and Disk Layout (Select Customize)

Customized Disks

1.	S0	/	512 MB
2.	S1	/var	2048 MB
3.	S2	(overlap)	(entire drive)
4.	S3	swap	2048 MB
5.	S4	/usr	1024 MB
6.	S5	/local	1024 MB (or rest of drive)
7.	S6	(unallocated)	(unallocated)
8.	S7	(unallocated)	(unallocated)

(Note: You probably want to leave some space (at least 20 megabytes) at the end of the drive and two slices available. These unused slices and space are left available because at a later date should you decide to install Solstice Disk Suite or Veritas Volume Manger you will not have to repartition and reinstall.
Mount Remote File Systems (No need, Select "Continue")

OS Installation Profile:

Look over these installation settings you have selected, if correct, select "Begin Installation".

Select the "Auto reboot" button and begin installation

Post OS Installation:

6. Settings Root's Password:

After the OS installation and the auto-reboot you should get to the login prompt. Log in as root and change root's password. Please choose a password that is both difficult to guess or crack but you will remember. This password should be at least eight characters and should include upper and lower case letters with

intermingled numbers and special characters (examples of special characters include: %,$,^,@,;,*,&,?) . Difficult passwords to crack are those in which brute-force methods are not successful or those when the computing time to brute-force a password is extremely long and impractical. Brute-force guessing/cracking is a method in which dictionaries or common passwords/ common letter sequences are tried in billions of combinations followed by generating every possible character combination either against your encrypted password or a program/network service that will show whether your password was successfully guessed.
More information on cracking passwords and brute-force methods can be found via:
http://www.cs.umn.edu/help/security/brute-force-cracking.html
http://www.itsecurity.com/asktecs/jul101.htm
http://www.openwall.com/john/
http://northernlightsgroup.hypermart.net/nutcracker.html

You should also never include sequences of numbers/letters that can be guessed from personal information including: a social security number, pet's/relatives names, or an employee id number. Carefully committing this password to memory with some sort of mnemonic phrase also prevents you from writing it down where it could be found by someone unauthorized. There are many references for choosing good passwords, I suggest finding and reading at least a couple of them.
http://web.mit.edu/network/passwords.html
http://home.netscape.com/security/basics/passwords.html
http://www.net.berkeley.edu/dcns/faq/good_pw.html
http://geodsoft.com/howto/password/password_advice.htm
http://law.utoledo.edu/it/badpass.html

7. Network Settings:

Create an /etc/defaultrouter file (**vi /etc/defaultrouter**) with the IP address of the router or firewall that is this machine's gateway to the rest of the network.
Create an /etc/notrouter (**touch /etc/notrouter**), so that this host itself will not behave as a router in the event that a second network interface is installed or activated. We do this for all hosts, even those that will not be connected to multiple networks. We do not want any routing services to be running and this keeps in.routed and in.rdiscd from starting at boot time.

Configure DNS:
a. Create an /etc/resolv.conf (**vi /etc/resolv.conf**) file with nameserver (DNS) and optional domain and search information.

/etc/resolv.conf:

domain corpname.com	(Your domain name)
nameserver 10.8.1.1	(Your first nameserver IP)

```
nameserver 10.8.2.1          (Your second nameserver IP)
nameserver 10.10.8.1         (Your third first nameserver IP)
search corpname.com secondcorpname.com (Your local domains)
```

b. Edit /etc/nsswitch.conf (**vi /etc/nsswitch.conf**) and add "dns" to the "hosts:" line. This allows DNS lookups for hosts that are not in this system's /etc/hosts file.

/etc/nsswitch.conf:

```
# /etc/nsswitch.files:
#
passwd:     files
group:      files
hosts:      files dns
ipnodes:    files
networks:   files
protocols:  files
rpc:        files
ethers:     files
netmasks:   files
bootparams: files
publickey:  files
# At present there isn't a 'files' backend for netgroup;  the system will
#   figure it out pretty quickly, and won't use netgroups at all.
netgroup:   files
automount:  files
aliases:    files
services:   files
sendmailvars:   files
printers:       user files
auth_attr:  files
prof_attr:  files
project:    files
```

c. reboot this host

8. Install Additional Sun Solaris Software:

Some Solaris available tools for enhancing security on Solaris 8 are not installed with the "Core System Support" software group; we will need to add them.

Mount the "**Solaris 8 Software Disk 1** of 2":

mount –F hsfs –o ro /dev/dsk/c0t6d0s0 /mnt

(assuming c0t6d0s0 is your cdrom device)

Add the network time daemon utilities and the utilities to display unix manual pages:

cd /mnt/Solaris_8/Product/
pkgadd –d . SUNWntpr SUNWntpu SUNWdoc
(Answer y/yes to the installation questions.

Umount the "**Solaris 8 Software Disk 1** of 2"

umount /mnt
Always change directory out of the /mnt directory (**cd /** or other location) or /mnt subdirectories before attempting to umount a CD. The OS will report the CD filesystem as "busy" and will not umount unless you are not in or not currently using that filesystem.

Add the system/process accounting packages, on-line solaris manual pages, and SUNWlibC (found on disk 2):
(SUNWlibC is required for the solaris man utilities to function properly)

Mount the "**Solaris 8 Software Disk 2** of 2"

mount –F hsfs –o ro /dev/dsk/c0t6d0s0 /mnt
(Assuming c0t6d0s0 is your cdrom device)
cd /mnt/Solaris_8/Product/
pkgadd –d . SUNWaccr SUNWaccu SUNWman SUNWlibC
(Answer y/yes to the installation questions)

Additional Optional Software:

At this time I also install the bash shell, and the bzip/gzip/zip utilities. Bash is an alternative shell similar to the normally available provided UNIX shells (sh/ksh/csh) that many users/administrators prefer. Bzip, gzip, and zip are alternative compression programs similar to the system default available compress utility. We can use these compression utilities on the backup files and maximize our available disk space.

pkgadd –d . SUNWbash SUNWbzipx SUNWgzip SUNWzip

9. Downloading and Installing Operating System Patches:

It is important to keep current with solaris patches. The recommended patch set should be downloaded from
ftp://sunsolve.sun.com/pub/patches/8_Recommended.zip
and checked for updates once a week. If the patch cluster includes updates for packages you have on your system, install them at your earliest convenience. I choose to download the patch cluster automatically once a week and compare its checksum against its previously downloaded patch cluster. If the checksum changes I know that a newer patch cluster has become available. For security and

file integrity reasons verify the checksum output with the CHECKSUMS file that can optionally also be downloaded from the ftp//sunsolve.sun.com/pub/patches directory. The current CHECKSUMS file contains three checksums that you can verify. You can use the solaris provided **sum** utility to verify two of the checksums and download and compile the **md5sum** utility to perform the third check/comparison. Md5sum is part of the GNU textutils software package and can be obtained from ftp://ftp.gnu.org/gnu/textutils/.

To compile and install md5sum utility we do the following steps on a trusted host:
(textutils was at version 2.0 at time of this document)
gunzip textutils-<version>.tar.gz
tar xvf textutils-<version>.tar
cd textutils-<version>
./configure
make
(You can now copy the md5sum binary from textutils-<version>/src directory, write it to CD and use it off the CD or install it on the secure backup server).

For new installations (before network is connected), download the patch cluster and write it to a CD on another trusted host. (Note: If you did not install the zip utility in the optional additional software section above, please unzip the file first and write its entire directory/file structure to CD)

a. Unmount and remove existing CD in drive if present (**umount /mnt**)
b. Mount patch CD (**mount –F hsfs –o ro /dev/dsk/c0t6d0s0 /mnt**)
c. Copy Solaris 8 recommended patches zip file to a temporary location. (**mkdir /local/patches; cp /mnt/8_Recommended.zip /local/patches/.**)
d. Unzip patches (**cd /local/patches; unzip ./8_Recommended.zip**)
e. Change directory to patch directory and install patches) (**cd /local/patches/8_Recommended; ./install_cluster**)
   You may see quite a few:
   "Installation of $patch failed. Return code $number" error messages when you install these patches.
   The common patch return code 8 means that this patch is for a package that is not installed on this system. The common patch return code 2 means that this patch or a newer patch for this software package as already been applied.
f. Remove patches from temporary directory
   (**rm –rf /local/patches/8_Recommened***)
   This will remove both the zip file and the uncompressed patches directory
g. Reboot the host.

10. Modifying and Removing Solaris Operating System Network Daemons and Services Startup Scripts:

Removing scripts:

First we will remove unnecessary network daemon startup scripts and operating system services. Many of the default services and scripts that are normally started at boot time are harmful to a host's security. You can choose to either remove or rename the scripts you wish not to be started at boot time. I choose to remove them so they are never accidentally run or re-linked and started at boot time. If you need one of them again, chances are you have access to another solaris host that has these files and you can easily recreate them.

Removing auto configuration startups:

rm /etc/init.d/sysid.net /etc/rc2.d/S30sysid.net
rm /etc/init.d/sysid.sys /etc/rc2.d/S71sysid.sys
rm /etc/init.d/autoinstall /etc/rc2.d/S72autoinstall

Removing Name cache server, RPC and NFS related startups:

rm /etc/init.d/nscd /etc/rc2.d/S76nscd
rm /etc/init.d/rpc /etc/rc2.d/S71rpc
rm /etc/init.d/nfs.client /etc/rc2.d/S73nfs.client
rm /etc/init.d/nfs.server /etc/rc3.d/S15nfs.server
rm /etc/init.d/autofs /etc/rc2.d/S74autofs
rm /etc/init.d/*cache* /etc/rc2.d/*cache*

Remove LDAP cache manager, PRESERVE, and sendmail from startups:

rm /etc/init.d/ldap.client /etc/rc2.d/S71ldap.client
rm /etc/init.d/PRESERVE /etc/rc2.d/S80PRESERVE
rm /etc/init.d/sendmail /etc/rc2.d/S88sendmail

Modifying scripts/services that we DO wish to have run at boot time:

Create our own inetsvc script, I called mine /etc/init.d/newinetsvc.
In the startup script you just need the following two lines:

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

vi /etc/init.d/newinetsvc (add above 2 lines)
chown root:sys /etc/init.d/newinetsvc
chmod 744 /etc/init.d/newinetsvc

Remove the original link so the original inetsvc is not started. Add the new symbolic link in the rc2.d directory so our newinetsvc will be started at boot time.

rm /etc/rc2.d/S72inetsvc
ln –s /etc/init.d/newinetsvc /etc/rc2.d/S72newinetsvc

This much smaller version of the script keeps services like DNS and inetd from starting even if configuration files exist for these daemons. Inetd controls many unix network services like rshd, ftpd, and telnetd, while historically available and useful are NOW considered very insecure. The network services that inetd provides are often not encrypted and if found running can provide hackers with many remote user and root exploits including buffer overflows and denial of service attacks.

11. Syslog:

Modify the /etc/init.d/syslog script. Add the –t option to syslogd so that syslog will only listen to syslog messages from this local host and not accept syslog messages from other servers. There is no need to accept messages from other servers and this prevents any type of attacks this syslog service would allow by not listening or accepting syslog messages on a network accessible interface.

To accomplish this syslog hardening:

Change the below line in /etc/init.d/syslog from: /usr/sbin/syslogd >/dev/msglog 2>&1 & to: /usr/sbin/syslogd –t >/dev/msglog 2>&1 &

We also want to have syslog log more information that it does by default
One of the security parameters we want syslog to log is failed logins and other authentication information.

To accomplish this we need to:

a. Create /var/log/authlog:
   touch /var/log/authlog
b. Set its permissions and its ownership:
   chmod 600 /var/log/authlog
   chown root:sys /var/log/authlog
c. Edit /etc/syslog.conf and add the following line. (syslog.conf configuration syntax demands a TAB separation between the described log service and its specified log file)
   vi /etc/syslog.conf

auth.info /var/log/authlog

Note: Solaris 8 now creates the /var/log/authlog file, but you still need to add the above configuration line in /etc/syslog.conf

12. Sendmail:

We already removed the sendmail startup script that starts sendmail as a daemon so we are no longer processing incoming e-mail or clearing the host's local mail queue. However, we still want to be able to process and forward local e-mail that is generated by this host so we add the following "sendmail –q" line to roots cron. We edit roots cron entries via the "crontab" command. **(crontab –e root**).

You may want to verify and set your editor to vi before running **crontab –e root**. **(EDITOR=/usr/bin/vi; export EDITOR**)

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

We handle processing e-mail via cron rather then running sendmail as a daemon because it makes the normal running process table smaller (easier to audit) and the service doesn't need to run continuously to perform its task.

13. Deleting/Modifying Default Solaris Accounts:

Removing UNIX logins that are never used is necessary to prevent security holes associated with system accounts. Typically an account or service is broken into that runs as a specific user and is used to gather information and/or escalate user privileges to become root. Deleting these accounts also makes the system (/etc/passwd, /etc/shadow) files easier to audit.

Delete user accounts that are unnecessary for this server with the **userdel** command:

userdel uucp
userdel nuucp
userdel listen
userdel lp
userdel nobody4

Several other default system accounts do not have a shell set. Set their shell to /dev/null to prevent any type of interactive login access.

passmgmt –m –s /dev/null adm
passmgmt –m –s /dev/null daemon
passmgmt –m –s /dev/null bin

passmgmt –m –s /dev/null nobody
passmgmt –m –s /dev/null noaccess

14. Cron:

Disable cron for all users besides root and sys. The cron process starts a daemon that executes commands at specified dates and times. The "at" program allows you to execute commands at a later time. The "root" and "sys" accounts are the only users that will be doing this. This restriction helps prevent processes that you don't expect to see or want running on this host.

Here is how to accomplish this:

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

Enable cron logging and view /var/cron/log daily for anomalies.
To enable cron logging make sure /etc/default/cron reads as:

```
CRONLOG=YES
```

This should be the default setting in solaris 8.

15. Hardening the Default TCP/IP Settings:

These are the SANS recommended TCP settings. Create a startup script that will set these at boot time. We accomplish this by creating a file in the /etc/init.d/ directory with a symbolic link from /etc/rc2.d/ directory that will start the script. I have also included a very brief description of the settings benefit(s).

touch /etc/init.d/nddsettings
chmod 744 /etc/init.d/nddsettings
chown root:sys /etc/init.d/nddsettings

It's recommended to have this script run after the rc2.d/S69inet script. Creating a S69nddsettings will accomplish this.

ln –s /etc/init.d/nddsettings /etc/rc2.d/S69nddsettings

Add the following lines to your newly created /etc/init.d/nddsettings file.

```
#!/sbin/sh

/usr/sbin/ndd -set /dev/tcp tcp_conn_req_max_q0 8192
/usr/sbin/ndd -set /dev/tcp tcp_ip_abort_cinterval 60000
# Helps with protection from SYN flood attacks

/usr/sbin/ndd -set /dev/ip ip_respond_to_timestamp 0
/usr/sbin/ndd -set /dev/ip ip_respond_to_timestamp_broadcast 0
/usr/sbin/ndd -set /dev/ip ip_respond_to_address_mask_broadcast 0
# Prevents ICMP queries that would give out network configuration

/usr/sbin/ndd -set /dev/ip ip_forward_directed_broadcasts 0
# Disallow route broadcast packets from being forwarded.

/usr/sbin/ndd -set /dev/ip ip_ignore_redirect 1
/usr/sbin/ndd -set /dev/arp arp_cleanup_interval 60000
# Decrease arp caches, helps stop some IP spoofing techniques.

/usr/sbin/ndd -set /dev/ip ip_send_redirects 0
# Prevent ICMP redirects, denial of service attacks.

/usr/sbin/ndd -set /dev/ip ip_forward_src_routed 0
# Prevent hacker from telling a packet which route it should take.

/usr/sbin/ndd -set /dev/ip ip_forwarding 0
# Another place we disable ip forwarding.

/usr/sbin/ndd -set /dev/ip ip_strict_dst_multihoming 1
# Strict IP restriction when multi-homing, do not route
# between network interfaces
```

Further explanations on these settings can be found by searching http://www.sans.org, http://sunsolve.sun.com, or viewing the nddconfig script that can be obtained via http://www.sun.com/blueprints/tools/nddconfig_license.html.

16. Hardening Kernel Settings:

Kernel settings on solaris are set by editing the /etc/system file (and rebooting)

Add these entries to /etc/system.

```
set noexec_user_stack=1
set noexec_user_stack_log=1
```

The first line helps to prevent buffer overflow attacks. The second line helps aid in logging the attempted buffer overflow attacks. Buffer attacks are those in which an attacker gives a program/network service more or different information then it is expecting, in hopes that the "extra" information will be put into memory of the executing program and executed. Typically these buffer overflows are used to accomplish a denial of service attack or gain access to an interactive login.

17. Additional Auditing and Logging:

Enable our process accounting packages that we already installed.
Edit the /etc/init.d/perf script and uncomment lines indicated in the script's instructions.

Automating the running of the process accounting:
You can either edit the sys user's crontab entry by editing the /var/spool/cron/crontabs/sys file directly or by doing a crontab –e sys. I prefer crontab –e sys as it does some important syntax checking and file protecting/locking of the configuration file.

Edit the user "sys" crontab **(crontab –e sys**) and add the following lines:

```
0,10,20,30,40,50 * * * * /usr/lib/sa/sa1
55 23 * * * /usr/lib/sa/sa2 –s 00:00 –e 23:59 –i 1200 –A
```

The first line takes a snapshot of this hosts state every 10 minutes and the second line produces a daily report. You may wish to change the date format in /usr/lib/sa/sa1 and /usr/lib/sa/sa2 to more than just the day of the month.

Instructions:

Changing these DATE lines in the sa1 and sa2 scripts.

```
from the existing:
DATE=`/usr/bin/date +%d`            #which outputs just the day of the month
```

```
to:
DATE=`/usr/bin/date +%m.%d`        #which outputs the month and day   OR
DATE=`/usr/bin/date +%Y.%m.%d` #which outputs the year, month, and day
```

will allow you to keep much more historical information by avoiding the default monthly overwrites of these files at the price of using more disk space for these process accounting/auditing reports.
This process accounting gives the administrator a good understanding of what processes are being run on this system, when they run, and what resources are being utilized while they are running. Once a baseline is established it becomes possible to look for abnormal activities on this server.
More important then generating this audit/process information is looking at it!! These daily reports are stored in /var/adm/sa/. Be sure look at the output via the "**sar**" command, verifying that nothing unusual or unauthorized is appearing.
More information can be found about how to interpret System Activity Reporter output via:
http://www.cert.org/security-improvement/implementations/i027.02.html
http://dennis_caparas.tripod.com/Configuring_sar_for_your_system.html
http://osr5doc.ca.caldera.com:457/PERFORM/running_sar.html
http://cs.felk.cvut.cz/~zemanekp/datasem/datasem.htm

Change the default permissions for file creation. We do this to verify that files are not being created that could be easily modified by other users.

To accomplish this, edit /etc/default/login, uncomment this line:

```
#UMASK=022
```

Log all failed login attempts, rather then just every fifth consecutive failure.
Edit /etc/default/login, uncomment and change this line:

From:

```
#SYSLOG_FAILED_LOGINS=5
```

To:

```
SYSLOG_FAILED_LOGINS=0
```

18. Creating the /etc/motd & /etc/issue

We need to create a message of the day file (/etc/motd) and an issue identification file (/etc/issue) that inform all users of their responsibilities and will help aid if

any legal action need be taken against unauthorized or improper activities. We do this by creating a /etc/motd and an /etc/issue file.

vi /etc/motd (insert the below text) &
vi /etc/issue (insert the below text)

This system is for the use of authorized users only. Individuals using this computer system without authority, or in excess of their authority, are subject to having all of their activities on this system monitored and recorded by system personnel.

19. Configuring Network Time Synchronization:

Keeping this server's date and time synchronized is very important. It may be necessarily for all hosts in which you wish to coordinate activities(via logfiles) to have their time synchronized. Coordinating log files becomes crucial if a break-in occurs and you wish to glean as much information as possible from system log files. This time synchronization is done via the xntpd daemon- Network Time Protocol daemon.

Edit /etc/inet/ntp.conf and add the following lines

```
server  10.8.2.3        #(your 1st  time server IP)
server  10.3.2.8        #(your 2nd time server IP)
server   10.10.2.4      #(your 3rd time server IP)
restrict default        ignore
restrict 10.8.2.3       nomodify noquery     # (your 1st  time server IP)
restrict 10.8.2.8       nomodify noquery     # (your 2nd time server IP)
restrict 10.10.2.4      nomodify noquery     # (your 2nd time server IP)
restrict 127.0.0.1      nomodify             # (your 2nd time server IP)
```

Be sure to replace the above IPs with your authorized (obtain permission from your network administrator) timeservers. The first three lines in the above configuration file are the timeservers this server is using for synchronization. The next line restricts all access to this host's time service. There is no need for other hosts on this network to query or modify this server's time, so restricting this is the correct task. The next lines are exceptions to the "restrict all" line and allow this host to accept /"believe" the information it queries from the specified timeservers. The last line is so that this server can query it self for purposes of administrator testing.

With the now present ntp.conf file the xntpd process will start next time you reboot.

Note: In the environment for this practical the above 3 time servers are internal (non-internet) dedicated GPS (global positioning system) network time devices at three

different physical locations. If you don't have access to a time system with this degree of reliability/redundancy, have this server use itself as a higher stratum(fall-back) pseudo-clock (timer server) to prevent it from being skewed by an inaccurate timer server. Adding the below lines to the top of the /etc/inet/ntp.conf file accomplishes this:

```
server 127.127.1.1
fudge 127.127.1.1 stratum 8
```

The 127.127.1.1 IP address is a special IP address that tells xntpd to use the local server clock as a time server.

20. Add Our Additional Drive and Mount all Filesystems as "logging"

a. We need to **format** and layout the partition table for this new drive.

```
# format
0. c0t0d0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /pci@1f,4000/scsi@3/sd@0,0
1. c0t8d0 <SEAGATE-ST173404LCV-4301 cyl 14087 alt 2 hd 24 sec 424>
          /pci@1f,4000/scsi@3/sd@8,0
```

b. Select second drive (drive 1 from above) and verify that the layout is what is desired.

My partition table:

```
partition> print
Current partition table (original):
Total disk cylinders available: 14087 + 2 (reserved cylinders)

 Part      Tag         Flag     Cylinders  Size          Blocks
 0       unassigned    wm       0          0         (0/0/0)          0
 1       unassigned    wm       0          0         (0/0/0)          0
 2       backup        wu       0 - 14086  68.35GB   (14087/0/0) 143349312
 3         -           wu       0 -     0  4.97MB    (1/0/0)         10176
 4         -           wu       1 - 14086  68.35GB   (14086/0/0) 143339136
 5       unassigned    wm       0          0         (0/0/0)          0
 6       unassigned    wm       0          0         (0/0/0)          0
 7       unassigned    wm       0          0         (0/0/0)          0
```

c. Create a ufs filesystem on this drive, we do this with the newfs command.
 # newfs /dev/rdsk/c0t8d0s4

d. Make a mount point for this drive. Since this is going to be the drive we are going to store our secure backups we should simply mount it as /backups.
 # mkdir /backups

e. Mount this new drive under its new mount point.
 # mount –F ufs /dev/dsk/c0t8d0s4 /backups

f. Add this filesystem line to /etc/vfstab

```
/dev/dsk/c0t8d0s4 /dev/rdsk/c0t8d0s4 /backups ufs 2 yes logging
```

The "logging" Mount Option prevents many filesystem checks (fscks) at boot time. This "logging" option on large filesystems could save you hours in recovery time should this server go down without cleanly unmounting its filesystems in a crash or kernel panic. It is recommended to mount all Solaris ufs filesystems with this logging option. Add the logging option to all our ufs filesystems defined in /etc/vfstab at this stage in the build.
The /etc/vfstab file will now look similar to:

#device #to mount to fsck #	device point	mount type	FS pass	fsck at boot	mount options	mount
fd	-	/dev/fd	fd	-	no	-
/proc	-	/proc	proc	-	no	-
/dev/dsk/c0t0d0s3	-	-	swap	-	no	-
/dev/dsk/c0t0d0s0	/dev/rdsk/c0t0d0s0	/	ufs	1	no	logging
/dev/dsk/c0t0d0s4	/dev/rdsk/c0t0d0s4	/usr	ufs	1	no	logging
/dev/dsk/c0t0d0s1	/dev/rdsk/c0t0d0s1	/var	ufs	1	no	logging
/dev/dsk/c0t0d0s5	/dev/rdsk/c0t0d0s5	/local	ufs	2	yes	logging
/dev/dsk/c0t8d0s4	/dev/rdsk/c0t8d0s4	/backups	ufs	2	yes	logging
swap	-	/tmp	tmpfs	-	yes	-

21. Compile and run the Fix-Modes program:

The fix-modes application is a program that provides additional security to default Solaris installations. It goes through the files installed by the solaris operating system and checks file and ownership permissions. It looks at devices, files and directories that the operating system installs and uses the /var/sadm/install/contents as a reference for which files are on the system. Its duties include changing permissions on any file security vulnerabilities associated with world or group writable permissions.
Download the source from ftp://ftp.wins.uva.nl/pub/solaris/fix-modes.tar.gz to the secure build workstation.

Build and install the application:

Uncompress the fix-modes-tar.gz file

gunzip fix-modes.tar.gz

Create a directory for the source as it does not create its own when un-tared.

mkdir fixmodes

Move the source into this new directory

mv fixmodes.tar fixmodes

Change directory into fixmodes directory

cd fixmodes

Untar the source

tar xvf fixmodes.tar

Compile the source:

make (This command will build the binaries)

Tar up this new source:

tar cvf /tmp/fixmodes-built.tar ../fixmodes

Write this fixmodes-built.tar file to a CD and mount it in our new secure backup server.

Install the software:

mkdir –p /local/tmp/
cp fixmodes-built.tar /local/tmp
cd /local/tmp
tar xvf fixmodes-built.tar
cd fix-modes
./fix-modes (Runs the application)

Any modifications fix-modes makes will be shown in the /var/sadm/install/contents.mod file. If needed, you can undo the changes made by this application by running fix-modes –u (undo).

Remove the application directory and tar file after running it.

rm –rf /local/tmp/*

22. Mount Filesystems in a More Secure Manner:

There are several reasons why we created several partitions for this host. Among them including: the ability to isolate operating system files, from disk intensive applications, from the system logging partition. This granularity of filesystems also allows us some security flexibility when mounting the filesystems. Add the following mount attributes to the following filesystems:

/usr	ro (read-only, The logging can be removed as most of the time this filesystem will not have live data being written.)
/var	nosuid
/local	nosuid
/backups	nosuid

We mount /usr read-only to prevent binaries from being replaced with trojan binaries or other undesired /usr filesystem modifications. The /usr partition can be remounted for patch upgrades or other needed modifications via the **mount –o remount, rw /usr** command. A reboot is required to mount /usr read-only again. Other partitions could be mounted read-only but it is an acceptable risk to have them writable as they regularly will have data written to them.

The addition of the nosuid attribute for /var, /local, & /backups is to prevent malicious Set User ID files from being created on these partitions. More on why SUID files can be dangerous later in this document.

/etc/vfstab will now look similar to:

#device #to mount to fsck #	device point	mount type	FS pass	fsck at boot	mount options	mount
fd	-	/dev/fd	fd	-	no	-
/proc	-	/proc	proc	-	no	-
/dev/dsk/c0t0d0s3	-	-	swap	-	no	-
/dev/dsk/c0t0d0s0	/dev/rdsk/c0t0d0s0	/	ufs	1	no	logging
/dev/dsk/c0t0d0s4	/dev/rdsk/c0t0d0s4	/usr	ufs	1	no	ro
/dev/dsk/c0t0d0s1	/dev/rdsk/c0t0d0s1	/var	ufs	1	no	nosuid,logging
/dev/dsk/c0t0d0s5	/dev/rdsk/c0t0d0s5	/local	ufs	2	yes	nosuid,logging
/dev/dsk/c0t8d0s4	/dev/rdsk/c0t8d0s4	/backups	ufs	2	yes	nosuid,logging
swap	-	/tmp	tmpfs	-	yes	-

23. Account Creation:

Create an account for the administrator account for purposes of remote maintenance: (This account will be used daily by the administrator)
useradd –c "Jason Christensen" –s /bin/bash –d /home/jason –m jason

Create an account that will facilitate receiving a host client backup files:
useradd –c "BACKUP bclient1" –s /bin/sh –d /backups/bclient1 –m bclient1

Create an additional account that will facilitate as the storage and restore account:
useradd –c "RESTORE rclient1" –s /bin/sh –d /backups/rclient1 –m rclient1

Note: For each client host you backup a unique bclient# and rclient# user will be created and used. This important separation of each client is needed to isolate and securely backup hosts in our environment.

Set the passwords for each of these new users. Be sure to generate secure passwords for these accounts.
passwd $username (where $username is each of our newly created accounts)

Change the default permissions on the home directories so that only that user has access to their home directories and files.
chmod 700 /home/jason /backups/bclient1 /backups/rclient1

24. Download and Install SSH:

The ssh package is the at the core of this backup server. It is providing the authentication and encryption of the backed up filesystems as they are being backed up or restored across the network. It also provides us with a secure encrypted "telnet" connection in which the administrator can remotely log in and audit the system. Previous to ssh the common way to transport files or log into a unix host was via the BSD (Berkely) 'r' commands and/or the standard unix ftp/telnet services. The 'r' commands, the ftp services, and the standard unix telnet network applications transmitted passwords and data in the clear and facilitated many break-ins/attacks.
The ssh FAQ is very helpful in understanding what features are included in this client/server application.
http://www.employees.org/~satch/ssh/faq/ssh-faq.html

Ssh is another software package that needs to be pre-built on a trusted host and written to a CD. Download the newest stable release from http://www.ssh.com. (At the time of this document's creation, 3.1.0 is the current version of ssh.) There are license requirements for commercial usage of this software so either confirm you fall under its usage guidelines or purchase a license. You can also use openssh (http://www.openssh.org), which doesn't have this license restriction but has additional compile/build/runtime dependencies.

Download, uncompress, and untar source:
gunzip ssh-<version>.tar.gz
tar xvf ssh-<version>.tar
cd ssh-<version>

Configure the source and build the binaries:

```
./configure     --prefix=/local \
                --without-ssh-agent1-compat \
                --disable-tcp-port-forwarding \
                --disable-X11-forwarding \
                --disable-suid-ssh-signer \
                --with-foreign-etc-dir=/local/etc
                --with-libwrap
```

The "./configure" options we are specifying are security and installation preferences. We are installing the binaries and manual pages in the /local directory hierarchy. We are not going to support ssh1 compatibility. We are not going to support TCP or X11 forwarding. This forwarding can be especially dangerous as it allows users to forward connections to other servers. In a secure

environment this forwarding has its useful applications but it should be limited to specific hosts that need this functionality. The disable-suid-ssh-signer prevents ssh from installing the ssh-signer binary with SUID access. We don't need this binary SUID and limiting SUID binaries is always a good idea. SUID/SGID binaries are programs that have a setting that tells the kernel to run the process with the user/group privileges of the group or file ownership, rather then the user that invokes the process. This SUID/SGID files have historically been difficult to successfully/sufficiently restrict for just their intended task and can often allow a malicious user to have access to something (filesystem/device) that they normally do not and are not intended to possess. The foreign-etc-dir specifies that we also want the ssh and sshd configuration files to be in /local/etc instead of the default /etc. The libwrap option will allow us to set up an access list of hosts that will be allowed to connect to this ssh2 service and deny all others. This tcp wrapper functionality is very powerful as it allows a system administrator to control access to many tcp services in a uniform manner.

After you run "./configure" you will get a "Configuration summary:". Verify that you have your desired options. Note, the location of the ssh configuration files will be in /local/etc/ssh2.

```
Configuration summary directories:

    Installation prefix.....: /local
    bin directory...........: /local/bin
    sbin directory..........: /local/sbin
    man directory...........: /local/man
    ssh2 etc directory......: /local/etc/ssh2
    PID-file directory......: default
```

Run the make command to build the executables.

make

Copy the make utility to our ssh directory because make is not available on our backup server and we will need it to install the ssh program:

cp /usr/ccs/bin/make ssh-<version> (directory)

Tar up the directory:

tar cvf ssh-sansbackup.tar ssh-<version> (directory)

Write this ssh-sansbackup.tar file to a CD and mount it in our new secure server.

Install the software:

mkdir /local/tmp
cp ssh-sansbackup.tar /local/tmp
tar xvf ssh-sansbackup.tar

cd ssh-<version>
make install
cd /
rm –rf /local/tmp/ssh*

25. Configuring SSHD:

You should now have ssh installed. You can do a quick listing on the system to view the ssh related files. To view all files in /local do a:

ls -alR /local

Since ssh is currently the only software installed in /local all files will be ssh related.

We next need to make some sshd configuration changes. Edit the /local/etc/ssh2/sshd2_config.

Uncomment the UserConfigDirectory line:

```
#     UserConfigDirectory          "/etc/ssh2/auth/%U"
```

Change it to:

```
      UserConfigDirectory          "/local/etc/ssh2/auth/%U"
```

We are specifying the UserConfigDirectory because we do not want user accounts on this backup host to have modification access to their authentication keys. The default location for these keys is a users home directory. A user's home directory is a directory that a user has write access.
The reason being that we want to lock down the type of authentication and the number of authentication keys is intuitive. Ssh by default is very powerful in the number of ways a user can control their authentication and we wanted to restrict this functionality. This is just one step in restricting this authentication. We will make several more throughout this document.

We already disabled TCP/X11 forwarding with our compile options, but we should also change the following lines to unmistakably shut off TCP/X11 forwarding:

From:

```
#     AllowX11Forwarding           yes
#     AllowTcpForwarding           yes
```

To:

```
      AllowX11Forwarding           no
      AllowTcpForwarding           no
```

Rhost authentication (Berkley r commands authentication) will also not be needed or desired so we will change the following lines:

From:

```
#     IgnoreRhosts              no
#     IgnoreRootRHosts          no
```

To:

```
      IgnoreRhosts              yes
      IgnoreRootRHosts          yes
```

We also want to set all hosts within ssh2_config that are allowed to connect to this ssh service. We do this via the AllowHost line.

Change it from:

```
#     AllowHosts                localhost, foobar.com, friendly.org
```

To:

```
      AllowHosts                127.0.0.1, 10.10.10.2, 10.10.10.3, 10.10.10.99
#     AllowHosts                localhost, bclient1.corp.com, bclient2.corp.com,
#                               admin-workstation.corp.com
```

We include the IP addresses of the servers rather then their host names, as there is no need to trust or be dependent on DNS for this access list. We also include their DNS names as a comment below their IP address to better document these host entries.

We want to explicitly state that the root user cannot log in remotely. We will create an administrator account for the interactive login access into this host that will only be allowed to log in from specific remote servers and then "su" to root for a better audit trail. This interactive administrator login access is also for the purposes of daily reviews of log files and ongoing system maintenance. Add the following line to the sshd2_config to specify root cannot log in directly to this host as root from remote servers.

```
      PermitRootLogin           No
```

Next, we want to make sure that sshd2 daemon starts at boot time so we will create a startup script.

Here is a simple sshd startup script that we will create in the /etc/init.d/sshd2 file.

```
#! /bin/sh
#
# start/stop the secure shell server

case "$1" in

'start')
        # Start the SSHD2 server
        if [ -f  /local/sbin/sshd2 ]; then
                echo "starting SSHD2 daemon"
                /local/sbin/sshd2 &
        fi
        ;;
 'stop')
        # Stop the SSHD2 server
        PID=`/usr/bin/ps -e -u 0|/usr/bin/fgrep sshd2|/usr/bin/awk '{print $1}'`
        if [ ! -z "$PID" ] ;  then
                /usr/bin/kill ${PID}  1>/dev/null 2>&1
        fi
        ;;
*)
        echo "Usage: $0 { start | stop }"
        exit 1
        ;;
esac
exit 0
```

Steps to get this script installed:

vi /etc/init.d/sshd2 (include the above lines)
chown root:sys /etc/init.d/sshd2
chmod 744 /etc/init.d/sshd2
ln –s /etc/init.d/sshd2 /etc/rc3.d/S80sshd2

Reboot the system to verify that sshd starts automatically.

Verify that sshd2 starts:
ps –ef |grep sshd2
The output will be similar to:

```
root   187   161  0 19:55:39 console  0:00 grep ssh
root   168     1  0 19:52:30 ?        0:00 /local/sbin/sshd2
```

26. Generate the SSH Keys:

We use public key authentication for many reasons.

a. We don't have to hardcode (store passwords in clear text in the backup script/process) or determine how to pass passwords securely to ssh in the backup script.
b. In many environments depending on source addresses (host-based authentication) would not be secure. Successful IP spoofing hacks are greatly deterred by not depending on a source's IP for absolute authentication. In the situation where a proxy firewall or other network device that may translate IP addresses (NAT) is in your network path, it is very possible that several hosts that we want as backup clients and other non-backup network entities could appear like they are coming from a single source IP address.
c. The public keys can be generated with much longer (more secure) passwords (phrases) then traditional passwords and in the case of our restore account we will create a pass phrase that would be nearly impossibly to guess/crack.

Generate ssh authentication keys for the user accounts that we previously created. The bclient1 account will have keys without a pass phrase to allow for our automated nightly backups that will run unattended out of our client's crontab.

a. Create a protected directory to temporarily store these keys.
mkdir /local/etc/ssh2/newkeys
chown root:root /local/etc/ssh2/newkeys
chmod 700 /local/etc/ssh2/newkeys

b. Create a key pair for the bclient1 user account.
/local/bin/ssh-keygen2 /local/etc/ssh2/newkeys/bclient1

When prompted for pass phrase leave it blank. You will see a warning message about creating keys with a "NULL passphrase".
We now have a bclient1 and a bclient1.pub key pair.

c. Do the same key creation for the administrator account and the rclient1 account. This time pick good pass phrases that would be difficult to guess or crack, but that you will remember. These pass phrases can be much longer in length then the standard unix passwords.
/local/bin/ssh-keygen2 /local/etc/ssh2/newkeys/jason
/local/bin/ssh-keygen2 /local/etc/ssh2/newkeys/rclient1

The administrator keys (jason.pub/jason) will be used by the administrator to log into this host over a secure encrypted ssh session and will allow daily maintenance and log checking/auditing to occur. These keys will need to be

securely copied to the administrator's workstation to facilitate this authentication/login process. This can be done via scp (ssh secure copy) once this host is connected to the network.

27. SSH Server Key Placement:

a. Create the UserConfigDirectory directory that we specified earlier to store these ssh keys. Specify directory ownership and directory permissions that will not allow other users access to these directories.

mkdir /local/etc/ssh2/auth
chown root:root /local/etc/ssh2/auth
chmod 700 /local/etc/ssh2/auth

b. Create user directories in /local/etc/ssh2/auth for each of the users.

mkdir /local/etc/ssh2/auth/jason
mkdir /local/etc/ssh2/auth/bclient1
mkdir /local/etc/ssh2/auth/rclient1

c. Copy each public key we previously created for each user into their own UserConfigDirectory.

cp /local/etc/ssh2/newkeys/jason.pub /local/etc/ssh2/auth/jason/
cp /local/etc/ssh2/newkeys/bclient1.pub /local/etc/ssh2/auth/bclient1/
cp /local/etc/ssh2/newkeys/rclient1.pub /local/etc/ssh2/auth/rclient1/

d. Create an "authorization" file that specifies the name of the public key to be used in each /local/etc/sshd2/auth/$username/ directory.

/local/etc/ssh2/auth/jason/authorization will contain:

Key jason.pub

&
/local/etc/ssh2/auth/rclient1/authorization will contain:

Key rclient1.pub

However, /local/etc/ssh2/auth/bclient1/authorization will contain:

Key bclient1.pub Command "/usr/bin/dd bs=64k of=$SSH2_ORIGINAL_COMMAND"

This "Command" argument is an implementation of ssh2 "Forced Commands" feature. Read the ssh2 documentation (man page) to further comprehend this feature (http://www.employees.org/~satch/ssh/faq/manpages/ssh2_man.html). A brief description of this feature is that it attempts to restrict specific authentication keys to executing just specific commands. There are many caveats

with this feature including the difficulty of really restricting what tasks a determined user can accomplish. The "Command" restriction used above nevertheless provides some user restriction and actually will help simplify the command syntax for doing client backups. The command syntax for client backups will be covered later in this document.

We will need both the bclient1 (private key) and the bclient1.pub (public key) later when we set up the backup client host.

28. Configure TCP_Wrappers:

TCP_Wrappers will allow or deny access from specific clients to our sshd2 service. Even though sshd2 was configured (above) to deny unauthorized hosts from authenticating this will deny the service earlier in the connection process and represents an example of security in depth. We will configure tcp_wrappers to allow just the clients that need to make a connection and deny all others.

Note, we will have to remember anytime we want to add a host access to sshd2 service on this host we will need to perform both steps. Adding it both to the tcp_wrapper configuration file and the sshd2_config configuration file.

Create/edit the tcp_wrapper configuration files:

vi /local/etc/hosts.allow
and add the following line (using your IP numbers):
localhost (127.0.0.1) is included so pre-network testing is possible.
10.10.10.2 & 10.10.10.3 are the IP numbers that clients that are being backed up for this exercise and 10.10.10.99 is the IP number of the workstation the administrator will be logging in from, to do daily maintenance and security related duties.

```
sshd2: 127.0.0.1, 10.10.10.2, 10.10.10.3 10.10.10.99
```

vi /local/etc/hosts.deny
and add the following line:

```
ALL: ALL@ALL
```

Note: In order for tcp_wrappers to be functioning in the above example it had to be built into the ssh2 program. This was done with the --with-libwrap option when we compiled this program.

29. Backup this Server:

Backup filesystems to the local tape drive. The following script can be used to backup the existing filesystems. Consider doing this to two tapes and having one tape at a secure off-site location.

```
#!/bin/sh
for filesystem in / /var /usr /local /backups
do
        /usr/sbin/ufsdump 0f /dev/rmt/0 $filesystem
done
```

30. Backup Software:

As any medium to large-scale backup server solution this server has an external tape library. This device has commercial software that comes from the tape library vendor that needs to be installed to allow the operating system to communicate with the tape library. Install this software (StoreageTek Timberwolf for this practical). You may also have commercial software to manage the many volumes of data on your DLT tapes. If so, try to install the software in standalone mode and verify that it isn't opening up any remote services. You can verify this functionality by using an external port scanner (like NMAP, more information on NMAP later in this document) and running **netstat –an** and looking for "LISTENING" applications. Shopping around for a software package and communicating with a vendor to verify that a product will manage you're large number of tapes/volumes and perform the backups of a large disk host without running any network accessible services is a recommendation for this build. For this practical the stand-alone backup software that was chosen was Legato Networker Workstation for Unix (http://www.legato.com). Reading the manuals and being very familiar with both backup features and disaster recovery methods is also recommended. The time to learn how to do disaster recovery is now, before a real problem arises.

31. Checking Configuration Before Network:

a. Check available network services
   netstat –a or
   netstat –a |grep LISTEN
Should show only sshd2 (port 22) listening

b. Verify permissions on any files or directories that were modified or created during installation. Since the main role of this server is to receive and store network backups verifying the permissions on the home directories of the rclient# and bclient# from time-to-time is a good idea.

ls -ld /backups/*

```
drwx------  2 bclient1 other      512 Jan 13 20:42 /backups/bclient1
drwx------  2 rclient1 other      512 Jan 13 20:42 /backups/rclient1
```

c. Check the logs, now and daily.

View /var/adm/messages and /var/log/authlog now and look for any warnings or errors.

32. Install Tripwire

Tripwire is a file integrity checker. If managed correctly it can alert you on all relevant filesystem changes. It creates a MD5 checksum database of files you wish to watch for tampering. Specifically it will help you detect both authorized and unauthorized changes to your filesystems. This document is not meant to teach you everything about tripwire, just simply a run through of a very basic setup. Please read the available documentation for further explanations. This practical will use the academic version of the software, please confirm you qualify for this version.
Download the source from http://www.tripwire.com/products/tripwire_asr/ to our secure build server. (Tripwire-1.3.1-1.tar.gz was current at time of document)
Uncompress and untar the souce

gunzip Tripwire-1.3.1-1.tar.gz
tar xvf Tripwire-1.3.1-1.tar.gz (This will create a tw_ASR_1.3.1_src directory)
cd tw_ASR_1.3.1_src
make (build the binaries)

Tar up the directory.

cd ..
tar cvf /tmp/tripwire-built.tar tw_ASR_1.3.1_src

Write this tripwire-built.tar to CD and transfer it to our secure backup server.

Un-tar the tar file on our server

cp /cdrom/cdrom0/tripwire-built.tar /tmp
cd /tmp
tar xvf tripwrite-built.tar
cd tw_ASR_1.3.1_src

Copy the binaries and man pages into place.

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

Note: Tripwire recommends that you store its integrity database on a read-only media such as a write-protected floppy or tape (Note, most sun servers do not have floppy drives). This is recommended because if the integrity database is compromised, tripwire will not alert on unauthorized filesystem changes.

Create a directory for the tw.config and copy the default tw.config file for this operating system into /usr/local/bin/tw

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

Initialize the database

cd /var/

tripwire –initialize (tripwire will create a /var/databases directory)

Several tripwire phases occur when you initialize the database, again, read the documentation.

The tripwire initialization creates a tw.db_sansbackup file containing checksums of the installed files.

Note: Again, the preferred place to store these tripwire databases is on read-only media.

Transfer a copy of this database to your local tape drive on a new blank tape.

tar cvf /dev/rmt/0 /var/databases

Eject the tape and write protect it.

When you update the tripwire database and copy the database to the tape (make the tape writable and then immediately write protect it again). You need to update the database when you make changes to the operating system (patches/additional software/etc). You update the database via the "**tripwire –interactive"** command.

Next we will set up set up a nightly restore of this write-protected directory for purposes of securely checking the filesystem for modifications. You may wish to also store the tripwire binaries on a CD and run them from non-writable media (the CD) to prevent any type of unauthorized "hacks" to these binaries. This nightly restore is done so that we know that the tripwire database itself has not been tampered. (The commercial version of tripwire has some added functionality for centralizing/streamlining this entire process and securely signing the databases.)

First create the nightly check script that will mail the administrator the tripwire reports.

/usr/local/bin/twcheck.sh:

```
#!/bin/sh
mt –f /dev/rmt/0 rewind  # rewind the tape
tar xvf /dev/rmt/0 # this restores a fresh copy of the /var/databases/tw.db_sansbackup
/usr/local/bin/tripwire –d /var/databases/tw.db_sansbackup | \
        mailx -s "Tripwire Report from sansbackup" $admin-emailaddress
        # Change to admin's real e-mail address
```

Steps in creating the above script:

vi **/usr/local/bin/twcheck.sh** (insert the above script)

chown root:root **/usr/local/bin/twcheck.sh**
chmod 700 **/usr/local/bin/twcheck.sh**

Second, add the below line to roots crontab (crontab –e root) to have this script run every night at 1am.

```
0 1 * * * /usr/local/bin/twcheck.sh
```

Resources on tripwire:
http://www.cert.org/security-improvement/implementations/i002.02.html
http://www.itworld.com/Sec/2308/IWD010604webtaggers/
http://www.itworld.com/Sec/2202/CED010416STO59611/
http://www.tripwire.com/files/downloads/asr/Tripwire-1.30-docs.pdf

33. Install LogSentry:

LogSentry is a program that will help flag interesting information that is logged to syslog files. Its report will show system activities in a daily report and should be used in addition to the daily viewing of the log files.
The software can be downloaded from http://www.psionic.com/products/logsentry.html.
The software is mostly shell script with the exception of the logtail binary. Build this logtail binary on the secure host.
Download the software
Un-compress and untar the source

gunzip logsentry-1.1.1.tar.gz
tar xvf logsentry-1.1.1.tar (This creates a logcheck-1.1.1 directory)
cd logcheck-1.1.1

Build the logtail binary with your C compiler

gcc –o logtail src/logtail.c

Tar up the directory.

cd ..
tar cvf /tmp/logsentry-built.tar logcheck-1.1.1

Write this logsentry-built.tar to CD and transfer it to our secure backup server.

Un-tar the tar file on our server

cp /cdrom/cdrom0/logsentry-built.tar /tmp
cd /tmp
tar xvf logsentry-built.tar
cd logcheck-1.1.1

Copy the logsentry program into place on our secure backup server.

cp systems/sun/logcheck.hacking /usr/local/etc
cp systems/sun/logcheck.violations /usr/local/etc
cp systems/sun/logcheck.violations.ignore /usr/local/etc
cp systems/sun/logcheck.ignore /usr/local/etc

cp systems/sun/logcheck.sh /usr/local/etc
cp logtail /usr/local/bin
chmod 700 /usr/local/etc/logcheck.sh
chmod 700 /usr/local/bin/logtail
chmod 600 /usr/local/etc/logcheck.violations.ignore
chmod 600 /usr/local/etc/logcheck.violations
chmod 600 /usr/local/etc/logcheck.hacking
chmod 600 /usr/local/etc/logcheck.ignore

Customizing /usr/local/etc/logcheck.sh:
First change the SYSADMIN in the logcheck.sh script

From:

```
SYSADMIN=root
```

To:

```
SYSADMIN=$emailaddress
```

Where $emailaddress is the system administrators e-mail address.

Edit the logcheck script and add authlog information (Earlier in this document we set up the authlog logging in the syslog.conf.)
Add the following line:

```
$LOGTAIL /var/adm/messages >> $TMPDIR/check.$$
```

below the existing configuration lines in/usr/local/etc/logcheck.sh:

```
# SunOS, Sun Solaris 2.5
$LOGTAIL /var/log/syslog > $TMPDIR/check.$$
$LOGTAIL /var/adm/messages >> $TMPDIR/check.$$
```

Add logcheck.sh to roots crontab so we will see pertinent report information hourly. (**crontab –e root** and add:)

```
0 * * * * /bin/sh /usr/local/etc/logcheck.sh
```

Customizing the /usr/local/etc: logcheck.ignore, logcheck.hacking, logcheck.violations, * logcheck.violations.ignore will allow you to specify what you want to see in these hourly reports.

A sample report can be found at http://www.psionic.com/products/logsentrysample.html. Please read the INSTALL & README files that come with this package as they help clarify further customization efforts of this software package.

34. Plug in the Network Cable:

Plug in your network cable and verify that you have network connectivity. Test routing, DNS, ssh connections again to other servers, network time, and outgoing sendmail. Test routing by running **traceroute** to several hosts verifying connectivity and that you take the correct network path. Test ssh by verifying that the administrator can log in remotely and perform daily responsibilities. Test network time by verifying this hosts time against another known accurate host. (**ntpq** –p can be helpful in verifying timeserver functionality) . Test sendmail by generating e-mail from this host to your e-mail address and by checking that you receive your logsentry and tripwire reports. (mailx –s "SUBJECT" $emailaddress can be used to quickly generate e-mail from this host)

At this time also verify that your solaris host is auto negotiating the correct network speed. Looking in /var/adm/messages and look at last entries from your network device (usually hme0). If you have to manually set your network card speed, "ndd "can be your friend. Ndd can also query the network device and can be helpful when troubleshooting network cards. More information on ndd can be found at http://sunsolve.sun.com.
Here is an example of ndd syntax to force an hme0 card to 100 Mbps full duplex:

```
/usr/sbin/ndd -set /dev/hme instance 0
/usr/sbin/ndd -set /dev/hme adv_100fdx_cap 1
/usr/sbin/ndd -set /dev/hme adv_100hdx_cap 0
/usr/sbin/ndd -set /dev/hme adv_10fdx_cap 0
/usr/sbin/ndd -set /dev/hme adv_10hdx_cap 0
/usr/sbin/ndd -set /dev/hme adv_autoneg_cap 0
```

It is similar syntax if you wish to add these lines to /etc/system so it happens at boot time:

```
set hme:hme_adv_100fdx_cap=1
set hme:hme_adv_100hdx_cap=0
set hme:hme_adv_10fdx_cap=0
set hme:hme_adv_10hdx_cap=0
set hme:hme_adv_autoneg_cap=0
```

35. Checking Configuration After Network Connectivity:

a. Run a port scanner against your server. Nmap is a good one and can be obtained from http://www.insecure.org/nmap/. Run this utility from another host.

To build Nmap download the source from the above link.

Un-compress,un-tar it, and build the source:

gunzip nmap-2.54BETA31.tgz
tar xvf nmap-2.54BETA31.tar
cd nmap.254BETA31
./configure
make

Run Nmap:

./nmap -sT -sU sansbackup will scan your server for both TCP and UDP available services and its output should be similar to:

```
Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
Interesting ports on  (sansbackup):
(The 3080 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
Nmap run completed -- 1 IP address (1 host up) scanned in 128 seconds
```

Note: Run "nmap" against your host from your workstation or other unix host on a daily or weekly schedule to verify that no additional network services have been started accidentally or maliciously. I suggested receiving the output daily even if it doesn't change, if for no other reason just to make sure the script is running and you would receive any significant changes.

Simple script that should be run daily via cron:

dailysansbackupnmap.sh:

```
#!/bin/ksh
/usr/local/bin/nmap sansbackup > /var/nmap/sansbackup.today.nmap
        #Choose an output directory (/var/nmap) that only the
        #user you are running the nmap has access to.
        #This is done so that no other user or process could write to that
        #directory or that known file output for malicious  purposes.
echo "**** Daily diff of nmap output ***** >/var/nmap/nmap.mail.output
diff /var/nmap/sansbackup.today.nmap /var/nmap/sansbackup.previous.nmap \
        >> /var/nmap/nmap.mail.output
echo "****************************** >> /var/nmap/nmap.mail.output
echo """>>/var/nmap/nmap.mail.output
echo "**** Nmap output from today ******" >> /var/nmap/nmap.mail.output
cat /var/nmap/sansbackup.today.nmap >> /var/nmap/nmap.mail.output
echo "******************************">>/var/nmap/nmap.mail.output
#Below, replace $YOUREMAIL with the e-mail address where you want
#this daily output to go
mailx –s "Daily nmap output" $YOUREMAIL < /var/nmap/nmap.mail.ouput
```

```
#Set up sansbackup.previous.nmap for tomorrow
cp /var/nmap/sansbackup.today.nmap /var/nmap/sansbackup.previous.nmap
```

Edit crontab on the workstation and have it run daily: (Everyday at noon is the below crontab example)

crontab –e $user and add the below line.

```
0 12 * * * /usr/local/bin/dailysansbackupnmap.sh
```

b. Verify the ssh access lists behave as desired. Attempt to connect to your host from several locations, both from authorized and unauthorized hosts. Verify that failed and successful login attempts are logged. Verify each user account that you created works and its permissions are set up correctly.
This testing is simply done by ssh-ing to this host from several machines and viewing the screen output of the test and the log entries in /var/log/authlog and /var/adm/messages.

c. Reboot the system and verify that your network connection and network speed come up correctly. Also verify again that sshd2 starts automatically.

Testing Functionality of Backups/Restores

36. Do a few client backups:

We have now set up a secure solaris backup server. Now we perform a few client-initiated backups. (ssh2 will be installed on this client host) Reading through the manual pages for dump/ufsdump and restore/ufsrestore is highly recommended.

On client:

a. Create a secure directory to store your authentication keys

mkdir /.ssh2
chown root:root /.ssh2
chmod 700 /.ssh2

b. Securely move your client keys (bclient, bclient.pub & rclient1 rclient1.pub) to your client and put them in your secure directory. It is advised that after you place these keys on your client, you remove them from the /local/etc/sshd2/newkeys directory. There is no need to keep these authentication keys in this location, and if misplaced fresh keys should be created.

c. Create a /.ssh2/identification file naming your authentication keys.

echo "IdKey bclient1" > /.ssh2/authentication
echo "IdKey rclient1" >> /.ssh2/authentication
chown root:root /.ssh2/authentication
chmod 400 /.ssh2/authentication

d. We now have a key authentication relationship between out client and server to backup our desired filesystems. We will use the solaris provided ufsdump and ufsrestore utilities. Most unix flavors provide dump and restore utilities so this should be mostly universal among any unix client hosts. For my example backups and restores both hosts are solaris 8 workstations. Below is a script that can be run interactively or run nightly from cron. Note: It is using the bclient1 account on the backup server to receive the backup files and using the bclient1 authentication keys on the client and server.

Client Backup Script: (run as root)

```
#!/bin/sh
# Back up script that will backup the / /var /usr & /opt filesystems to
# our sansbackup backup server.
# Suggest using IP numbers instead of DNS name to prevent DNS
# related security issues with transferring these filesystem files

DATE=`/usr/bin/date +%Y.%m.%d`
BACKUPSERVER="sansbackup.corpname.com"

/usr/sbin/ufsdump 0cf - / |/local/bin/ssh2 ssh2 –l bclient1 \
        $BACKUPSERVER bclient1.root.$DATE
echo "done" | /local/bin/ssh2 –l bclient1 bclient.root.$DATE.done'

/usr/sbin/ufsdump 0cf - /usr |/local/bin/ssh2 –l bclient1 \
        $BACKUPSERVER bclient1.usr.$DATE
echo "done" | /local/bin/ssh2 –l bclient1 $BACKUPSERVER \
        bclient.usr.$DATE.done'

/usr/sbin/ufsdump 0cf - /var |/local/bin/ssh2 –l bclient1 \
        $BACKUPSERVER bclient1.var$DATE
echo "done" | /local/bin/ssh2 –l bclient1 $BACKUPSERVER \
        bclient.var.$DATE.done'

/usr/sbin/ufsdump 0cf - /opt |/local/bin/ssh2 –l bclient1 \
        $BACKUPSERVER bclient1.opt.$DATE
echo "done" | /local/bin/ssh2 –l bclient1 $BACKUPSERVER \
        bclient.opt.$DATE.done'
```

Note: The full command line syntax for the above backups would include the "/usr/bin/dd bs=64 of=" if it were not part of the "Forced Command" ssh authentication process that we defined for the bclient user earlier in this document. Example: The last section in the above script, where /opt is being backed up, the syntax *would* have been:

```
/usr/sbin/ufsdump 0cf - /opt |/local/bin/ssh2 –l bclient1 \
        $BACKUPSERVER /usr/bin/dd bs=64 of=bclient1.opt.$DATE
/local/bin/ssh2 –l bclient1 $BACKUPSERVER echo "done" > \
        bclient1.opt.$DATE.done
```

37. Move Backup Files:

Moving the backup files from the backup user account on the server to the restore/storage account on the server should be done periodically. The above script will create a $filename.done file that will let us know with the backup is complete. When the backup is complete, move the backup file from the bclient directory to the rclient directory. At this time we will change the backup file's ownership to the rclient user and set it's permissions. Remove the $filename.done file when this is complete.

Manually:

mv /backups/bclient1/bclient1.$filesystemfile /backups/rclient1/
chown rclient /backups/rclient1/bclient1.$filesystemfile
chmod 600 /backups/rclient1/bclient1.$filesystemfile
rm –f /backups/bclient1/bclient1.$filesystemfile.done

or

Create a script to run out of root's cron:

```
#!/bin/ksh
for backupdone in /backups/bclient1/*.done
do
mv ${backupdone%.done} /backups/rclient1/.
chown rclient  /backups/rclient1/*
chmod 600 /backups/rclient1/*

rm –f $backupdone
done
```

This can be done via root's crontab automatically several times an hour. This process should be done often so that backup files are not kept unnecessarily long in the bclient1 directory. The rclient1 account was designed to store these filesystems and has

an additional security step that is in place with login access to this account. (Possession of the rclient keys and the authentication pass phrase is this additional security step).

Optionally you can bzip2 these files in the rclient directory to save disk space.

38. Do a Restore:

Go to the client directory mount point in the filesystem of the desired file or directory your restoring. This is if you want to restore the file/directory in its original location. Remember that the files are stored in the ufsdump file relative to their mount point. If you want to restore this file/directory in an alternative location, remember that the files/directories will restore in relative pathname from which they were saved/ufsdumped. Read the man page of ufsrestore for more information.

If your restoring the file "/var/log/authlog.2" and the filesystem is mounted as /var go to the /var directory and perform the restore. Below are two examples of restoring this file, one using ufsrestore in a non-interactive way and one interactively.

Non-Interactive ufsrestore (on client):
cd /var
**/local/bin/ssh2 –l rclient1 sansbackup cat bclient1.var.$date **
| /usr/sbin/ufsrestore xf - log/authlog.2
(prompted for ssh authentication pass phrase for the rclient key pair)

Note: "/var/log/authlog.2" is just "log/authlog.2" in the above command because of the relative pathname.

Interactive ufsrestore (on client):
cd /var
**/local/bin/ssh2 –l rclient1 sansbackup cat bclient1.var.$date **
| /usr/sbin/ufsrestore if –
(prompted for ssh authentication pass phrase for the rclient key pair)
ufsrestore > **cd log**
ufsrestore > **add authlog.2**
Warning: ./log: File exists
ufsrestore > **extract**
set owner/mode for '.'? [yn] n
Directories already exist, set modes anyway? [yn] n
ufsrestore > quit

Note: If you choose to compress the backup files with bzip change the above "cat" commands to "bzcat".

39. Ongoing Tasks:

This backup server must be watched closely. Check for Solaris security/patch updates and installed software updates. Look at and carefully view log files daily. Verify that you are receiving your daily reports and carefully reading them!! Each time you add a client host to be backed-up verify that its files are going into the proper directories, and that the directory and file permissions are correct. Document the procedure for adding clients/restoring files and share this information with your trusted administrators. It is optional whether you want all administrators to know the rclient pass phrases but make sure that key people (pun intended) are able to successfully restore files. For additional security consider encrypting and signing the ufsdump files on the server, but be aware this will add additional steps in verification and authentication before using these backed-up filesystems. This encryption and signing of the files can be done with GNU Privacy Guard (http://www.gnupg.org) package.

References

Pomerantz, Hal Solaris Security Step by Step v. 2.0, Sans Institute, 2001

Barrett, Daniel SSH The Secure Shell O'Reilly & Associates, February 2001.

Manis, Mark H. The UNIX Shell Programming Language Howard & Sams Co., 1986

Winsor, Janice. Solaris Advanced System Administrator's Guide Second Edition Macmillan, February 1998.

Pomeranz, Hal and Lee Brotzman, Securing Unix Systems, Sans Institute, October 2001

Suess, Jack. Unix Backup and File Restoration
URL: http://umbc7.umbc.edu/~jack/course/backups.html

Snailbook.com FAQ
URL: http://www.snailbook.com/faq/

SSH.FI Product FAQs
URL: http://www.sshi.fi/faq/

Satch, The Secure Shell Frequently Asked Questions
URL: http://www.employees.org/~satch/ssh/faq/ssh-faq-5.html#ss5.1

Noordergraaf, Alex Solaris Operating Environment Minimization for Security: A Simple, Reproducible and Secure Application Installation Methodology – Updated for Solaris 8 Operating Environment
URL: http://www.sun.com/blueprints/

Spitzner, Armoring Solaris Preparing solaris for a firewall, Updated August 19, 2001
URL: http://www.enteract.com/~lspitz/armoring.html

Solaris System Configuration
URL: http://info.rutgers.edu/Techdir/solarisbody.html