



UNIX Security Guideline

v1.0

May 29, 1998

*National Aeronautics and Space Administration
Office of Headquarters Operations
Information Technology and Communications Division*

UNIX Security Guideline

Table of Contents

1. Introduction.....	1
2. Purpose & Scope.....	1
3. Applicability.....	2
4. Policy.....	2
4.1 Passwords.....	2
4.2 Accounts.....	2
4.3 Reporting.....	2
4.4 Security Monitoring Activities.....	3
5. Practices.....	3
5.1 Patches.....	3
5.2 Network Security.....	3
5.2.1 Filtering.....	3
5.2.2 'R' Commands.....	5
5.2.3 /etc/hosts.equiv.....	5
5.2.4 \$HOME/.rhosts.....	6
5.2.5 NFS.....	6
5.2.6 /etc/hosts.lpd.....	10
5.2.7 /etc/ttytab (BSD UNIX systems) /etc/default/login.....	10
5.2.8 /etc/inetd.conf.....	10
5.2.9 Trivial ftp (tftp).....	11
5.2.10 /etc/services.....	11
5.2.11 tcp-wrapper.....	11
5.2.12 /etc/aliases.....	11
5.2.13 Simple Mail Transfer Protocol (SMTP).....	12
5.2.14 Majordomo.....	15
5.2.15 Fingerd.....	15
5.2.16 UUCP.....	15
5.2.17 WWW Security.....	16
5.3 Ftpd & Anonymous Ftp.....	19
5.3.1 Versions.....	19
5.3.2 SITE EXEC.....	19
5.3.3 Configuration of Your Anonymous Ftp Server.....	20
5.4 Password & Account Security.....	24
5.4.1 Proactive Checking.....	24
5.4.2 Root Password.....	24
5.4.3 NIS and /etc/passwd Entries.....	24
5.4.4 Password Shadowing and C2 Security.....	25
5.4.5 Administration.....	25
5.4.6 Special Accounts.....	25

5.4.7 Root Account.....	26
5.5 File System Security.....	26
5.5.1 General.....	26
5.5.2 /etc/rc.local.....	26
5.5.3 /usr/lib/expreserve.....	27
5.5.4 External File Systems/Devices.....	27
5.5.5 File Permissions.....	27
5.5.6 Files Run By Root.....	28
5.5.7 Bin Ownership.....	28
5.5.8 Tiger/COPS.....	28
5.6 SunOS Security.....	28
5.6.1 IP Forwarding.....	28
5.6.2 Framebuffers /dev/fb.....	29
5.6.3 /usr/kvm/sys/*.....	30
5.6.4 /dev/nit (Network Interface Tap).....	30
5.6.5 System Initialization.....	31
5.6.6 System Administration.....	32
5.6.7 Network Security.....	37
5.6.8 User Security.....	38
5.6.9 Audit Files.....	39
5.7 IRIX Security.....	39
5.7.1 /usr/lib/vadmin/serial_ports.....	40
5.7.2 IRIX Patches.....	40
5.7.3 IRIX Configuration Issues.....	40
5.8 X Windows Security.....	44
5.8.1 Problems With xdm.....	45
5.8.2 X Security -Tools.....	45
Appendix A - Security Tools.....	1
Appendix B - References.....	1
Appendix C - Shell Scripts and Useful Commands.....	1
Appendix D - xauth: How it works and Setup Instructions.....	1
Appendix E - Abbreviated Checklist.....	1

1. Introduction

The goal of this document is to update the existing UNIX Guideline with newer and more recent information. There is a wide variety of UNIX flavors in existence at NASA Headquarters and we have attempted to include specifics about some of them. This is a document that will never stop growing and changing so any input is very well appreciated.

To help you understand the format of the document we include the following details:

file names	<i>displayed in italics</i>
commands and daemon/processes	<i>displayed in bold italics</i>
File contents and host/domain names	displayed bold
CERT Advisories and ftp/http sites	<u>displayed in bold italics underlined</u>

2. Purpose & Scope

This UNIX-specific document augments existing NASA Headquarters automated information security (AIS) policies. This guide is addressed separately from other NASA AIS policies due to the prevalent threat to UNIX system security and the complex nature of UNIX systems.

This document is intended to focus on practices and procedures that can be implemented readily by NASA HQ System Administrators. It is understood that given the nature of the various UNIX operating systems and the functions being supported at HQ, the System Administrator is in a unique position to tailor the features of the UNIX system to: Meet the requirements of NASA functional managers. Safeguard the security of the information processed by the system. Safeguard the integrity of the NASA HQ domain. Safeguard the availability of NASA HQ resources for the appropriate personnel.

This document is intended to be dynamic and change with the changing UNIX computing environments and the changing security related technologies. The procedures and practices in this document, therefore, will change in response to new threats and vulnerabilities. The effective implementation of a UNIX security program will depend on:

- Implementation of the UNIX guideline.
- Periodic examination of the dynamic UNIX security practices to determine their pertinence and effectiveness to specific UNIX environments.

3. Applicability

This Guide applies to NASA HQ personnel, contractors, consultants, and any other personnel who use NASA HQ automated information systems. It also applies to owners and administrators of UNIX systems and to personnel using UNIX-like services on personal computers or Macintosh computers.

4. Policy

4.1 Passwords

Account passwords implemented on UNIX systems managed by NASA HQ shall comply with the NASA HQ Password Policy. Cases where policy cannot be complied with due to technical limitations of systems will be brought to the attention of the HQ ITS Manager for consideration as an exception to policy.

4.2 Accounts

1. Each user shall have their own account on each machine to which they require access. Assigned account holders are responsible for all activity under their accounts and shall not permit use by other personnel.
2. Accounts belonging to personnel who are away from the organization present a vulnerability. Individual users shall notify their System Administrators if their account will be inactive due to their absence. If the user does not intend to use their account for a period greater than one month, they shall coordinate with the System Administrator to have their account disabled and their e-mail forwarded to an appropriate point.
3. Accounts belonging to personnel who have departed NASA HQ shall be deleted upon the notification to the System Administrator of the departure. Prior to deleting accounts, the System Administrator shall ensure that system activities are not dependent on the account to be deleted. The NASA HQ Service Center is responsible for ensuring System Administrators are notified of the departure of personnel in their respective organizations.

4.3 Reporting

In the event of a real or suspected security incident, the Information Technology Security Team has developed an *Incident Response Process* that needs to be followed. This document outlines the User/System Administrator's roles and responsibilities, the Boeing IT Security Teams responsibilities, and the roles and responsibilities for the NASA IT Security personnel. The document can be found on the following NASA Web site:

www.hq.nasa.gov/security

4.4 Security Monitoring Activities

The HQ ITS Manager and Systems Administrators shall work in concert on the implementation of tools for monitoring and detecting security related activities of UNIX systems in the Headquarters domain. A list of tools available through the NASIRC is included in Appendix A.

The *Practices* in Section 5 and the corresponding abbreviated checklist in Appendix E shall be used by the HQ ITS Manager, support contractor security group and System Administrators to form the basis for an agreed upon set of tailored practices to be implemented on individual UNIX systems. The tailored practices shall be evaluated during the system certification process conducted by the HQ ITS Manager at least every three years on each UNIX system.

5. Practices

The substance of this section is based on a set of UNIX Security Guidelines and a security checklist prepared by the Australian Computer Emergency Response Team (AUSCERT). The Practical UNIX and Internet Security by Garfinkel & Spafford was utilized.

5.1 Patches

Install any patches not yet installed that are recommended for your system. The latest patch list may be available from your vendor or through the NASA HQ ITS Manager. Some patches may re-enable default configurations.

Therefore, it is important to go through this list of practices after installing any new packages. Verify the checksum information to confirm that you have retrieved a valid copy of the patch software. This is important in any anonymous ftp transaction. If checksums are provided - use them! That said, it should be noted that checksums produced by Abin/sum should not be considered secure. Other more cryptographically strong methods such as MD5 and the Secure Hash Algorithm (SHA) should be used.

5.2 Network Security

The features listed below can be used to help safeguard UNIX systems from attacks from external sources.

5.2.1 Filtering

Check whether the following ports are required outside your domain. If not required, then filter the ports out at the router. (Refer to C.6, Firewalls and Internet Security and to C.7, CERT Advisory CA-95.01 and CERT CA-96.21)

Name	Port	Protocol	Name	Port	Protocol
systat	11	TCP	login	513	TCP
netstat	15	TCP	shell	514	TCP
bootp	67	UDP	printer	515	TCP
tftp	69	UDP	biff	512	UDP
link	87	TCP	who	513	UDP
supdup	95	TCP	syslog	514	UDP
sunrpc	111	TCP/UDP	UUCP	540	TCP
NeWs	144	TCP	route	520	UDP
snmp	161	UDP	openwin	2000	TCP
xdmcp	177	UDP	NFS	2049	UDP/TCP
exec	512	TCP	x11	6000-6020	TCP

Also filter inbound packets showing a source IP address from within your internal network.

The following is a brief description of some of the ports:

- **systat (port 11)** Provides status information about our computer to other systems on the network.
- **tftp (udp port 69)** Is a UDP file transfer program with no security.
- **Sunrpe (udp and tep ports 111)** Portmapper, it mediates communications between network clients and servers that may have security problems.
- **SNMP/Simple Network Management Protocol (udp ports 161 and 162)** Designed to allow the remote management of devices on your network.
- **rexec (tep port 512)** Allows users to execute commands on other computers without having to log into them. Same vulnerability as ftp and telnet, the password is transmitted over the network unencrypted and it's susceptible to snooping.
- **UUCP (tcp port 540)** Used because some UUCP systems can transmit Usenet news with more efficiency than NNTP. Same risk as some of the others unencrypted passwords over the network.
- **X11/X Window System (tep ports 6000-6063)** Network based window system that allows many programs to share a single graphical display.
- **exec (port 512)** It was a Sun RPC server that allows for remote programs execution.
- **Syslog (udp port 514)** Used to manage log messages in a centralized way. Used in network devices like routers to report status and usage information. These devices do not have writable storage media so something has to be listening to their syslog messages. Hackers take advantage of this by flooding the syslog server until it runs out of space and stops logging anything new. This way evidence of the
- attacker activity is no longer logged and its lost.

5.2.2 'R' Commands

- a. Disable all 'r' commands (*Rlogin*, *rsh* etc.) unless specifically required. These commands use well known ports. *rlogin* uses port 513, and *rsh*, *rcp*, *rdump*, *rrestore*, and *rdist* which are different clients for the same server, use port 514.

Allowing the use of these commands may increase your risk of password exposure in network sniffer attacks. 'Y' commands have been a regular source of insecurities and attacks. Disabling them is by far the lesser of the two evils.

- b. Filter ports 512, 513, and 514 (TCP) at the router if you use any of the "r" commands. This will stop people outside your domain from exploiting them but will not stop people inside your domain from using them.

- c. To disable the "r," commands place a # in front of the line containing the service in the `letclinetd.conf` file, this will comment the line out. Then issue the following command:

```
lbinlps -aux | lusrbinlgrep inetd | lusrbinlgrep -v grep # lbinlkill -  
HUP <netd-PID>
```

This will send a HUP(Hangup or stop running) signal and will get *inetd* to reread its configuration file.

5.2.3 /etc/hosts.equiv

Check that this file does not exist. It contains a list of trusted hosts for your system. Any host listed in this files is considered trusted. Programs such as *rlogin* can then be used to log on to the same account name on a trusted machine without supplying a password. If you must have such a file:

- Ensure that you keep only a small number of trusted hosts listed.
- Trust hosts within your domain or under your management.
- Use *netgroups* for easier management if you run NIS (also known as 'YP').
- Only place fully qualified hostnames, no machine names /username pairs, i.e., `Hostname.hq.nasa.gov`.
- Ensure that you do not have a Y entry by itself anywhere in the file. The Y has the effect of making every host a trusted host.
- Ensure that the first character of the file is not J. [CERT Advisory CA-91:12](#).
- Ensure that the permissions are set to 600.
- Ensure that the owner is set to root.
- Check it again after each patch or operating system installation.

5.2.4 \$HOME/.rhosts

a. Check that no user has a `rhosts` file in their home directory. They pose a greater security risk than `/etc/hosts.equiv`, as one can be created by each user. This file allows each user to build a set of trusted hosts applicable only to that user. There are some genuine needs for these files, so hear each one on a case-by-case basis; e.g., running backups over a network unattended.

b. Use cron to periodically check for, report the contents of and delete `$HOME/.rhosts` files.

Cron allows you to schedule programs for periodic execution. Users should be made aware that you regularly perform this type of audit. An example of a cron entry follows:

```
10 0 *** find / -name .rhosts 2>& 1 > /usr/ladm/.rhost.log
```

This entry runs the `find` command at ten minutes after midnight to find all `.rhosts` files and send the results to the file called `rhost.log` under `lusrldm`.

If you must have such a file take into consideration the following:

- Ensure the first character of the file is not '-'. Refer to Appendix B item CERT Advisory CA-91:12.
- Ensure that the permissions are set to 600.
- Ensure that the owner is the same person who owns the account.
- Ensure that the file does not contain the symbol "+" on any line.
- Always use fully qualified domain names; e.g., `hq.nasa.gov`
- Do not include comment characters (`#` or `!`), these can create vulnerabilities.

5.2.5 NFS

a. Disable NFS if you do not need it. Check your vendor supplied documentation for details on how to do this, or contact the HQ AIS Manager for assistance. If you are going to use it, limit the machines to which file systems are exported. Also limit the number of file systems that each of the client machines mounts.

b. Apply all available patches; NFS has had a number of security vulnerabilities. If you are running NFS, make sure you are up to date on vendor patches and bug fixes. Particularly:

- Ensure the version of RPC portmapper does not allow proxy requests and that your system is not in the export list for a partition. This can cause a faked packet sent to your RPC system to fool your NFS system into acting as if the packet was valid and came from your own machine (spoofing).
- Ensure that your version of NFS does not allow remote users to issue the `mknode` command on partitions they import from your servers. If a user can create a new `/dev/kmem` device file

on your partition he will gain access to the kernel's memory. Reading or writing to the `/dev/kmem` device file accesses the kernel's memory.

- Be sure that your NFS version does the right thing when a user does a `cd..` in the top level of an imported directory from your server. Some older versions of NFS returned a file handle to the server's real parent directory instead of the parent to the client mount point. Assure that your server parses the export option list correctly. Old and more current NFS versions get accesses mixed up.

c. Filter NFS traffic at the router. Filter TCP/UDP on port 111 and on port 2049. This will prevent machines not on your subnet from accessing file systems exported by your machines.

d. Enable NFS port monitoring (portmon variable). This will cause any NFS requests that are not received from privileged ports to be ignored. Calls to mount a file system will then only be accepted from port < 1024 (root access). Check your vendor's documentation to see if this is an option for your version of UNIX. The way some NFS servers restrict this is by setting the kernel variable `nfs_portmon` to 1. Under SunOS, to enable NFS port monitoring add the following commands to `/etc/rc.local`:

```
/bin/echo"nfs_portmon/Wl"|/bin/adb-w/vmunix/dev/kmem> /dev/null 2>&l  
rpc.mountd.
```

Also using the adb debugger in SunOS:

<pre><i># adb -k -w /vmunix /dev/mem nfs-portmon/Wl _nfs_portmon: _nfs_portmon: 0 ?Wl \$q #</i></pre>	<pre>Changes kernel disk file Changes running kernel The default setting Change to 1 Write the result out</pre>
---	---

With Solaris 2.1-2.4 this can be done by inserting the following on the `/etc/system` file:

```
set nfs:n/s_portmon = 1
```

With Solaris 2.5 this can be done by inserting the following on the `/etc/system` file:

```
set n/ssrv:n/s_portmon = 1
```

e. Use `/etc/exports` and `/etc/dfs/dfstab` only to export file systems you need to export. If you are not certain that it needs to be exported, then it probably does not.

f. Run *fsirand* for all your file systems. This program makes it more difficult to guess a valid file handle. It randomizes the generation number of every Mode on a filesystem. *Fsirand* is not available in all versions of UNIX. First, check that you have installed any patches for *fsirand*. Older versions of Sun's *fsirand* contained buggy code that made the "random" values quite predictable. Then ensure the file system is unmounted and run *fsirand*. Predictable file handles assist crackers in abusing NFS. For example, to run *fsirand* on */dev/sd1a*:

```
# umount /dev/sd1a      Unmount filesystem /dev/sd1a
# /srand /dev/sd1a     Run fsirand
```

g. Check that you never export file systems to the world. Use a *-access=host.hq.nasa.gov* option in */etc/exports*.

h. Export file systems read only (option *ro*) whenever possible. By doing this the security and reliability of the file system is improved. It prevents the file systems from being modified by NFS clients, controlling the damage that can be caused by hackers, wayward users, and buggy software. File systems that should be read only:

- File systems containing applications
 - Organizational reference matter, policies and documents
- Netnews (if not using NNTP) Review the manual pages for "*exports*" An example,


```
/usr      -access=hq.nasa.gov -ro
```

i. Use the *secure* option if it is available on your version of UNIX. The biggest problem with NFS, configured normally, is the use of Sun's AUTH - UNIX RPC authentication system. AUTH-UNIX only needs that the user provide a UID and a group of GID's with every request. The NFS server trusts that the users are who they claim to be. With the use of secure NFS AUTH - DES RPC will be used to authenticate. Users must be able to decrypt a special key stored on the NIS or NIS+ server before NFS will allow the user to access files on that server. To specify secure NFS use the *secure* option as follows:

```
Server:
share -F nfs -o secure, rw=clients      /Usr
```

```
Clients: /etc/vfstab
#device device mount    FSfack mount mount
#to moun to fsck point  type pass at boot options
#
server:/Usr -      /Usr      nfs      - yes      secure
```

j. Use *showmount -e* to make sure you are exporting only the file systems you intended to export to the correct hosts with the correct options. This is a very useful command to verify proper

configuration of your NFS servers. The host option will allow you to view other servers export list. An example of the output of the *showmount -e* command follows:

```
% /usr/etc/showmount -e conch. navsea. navy.mil
export list for conch. navsea. navy.mil:
/usr          -access=navsea.navy.mil
/usr2/macro   -access=navsea.navy.mil
/usr3/public  -access=navsea.navy.mil
/var          -access=navsea.navy.mil
```

k. Check that the files and directories to be exported are owned by root. NFS serve maps root to nobody. Because of this you can protect your files and directories by setting their ownership to root and their protection mode to 755 (for programs and directories) and 644 (for data files). Clients will not be able to modify the contents of these. Information that clients need no access to should be set to 700 (for programs and directories) or 600 (for data files). Better yet, put them on a separate server. Remember, this will only protect files that are owned by root and if you make sure your kernel is not patched to set the value of nobody to 0, or if you export the filesystem to a particular host with the *- root=* option.

l. Check that */etc/exports*, is owned by root and has permissions of 644.

m. Do not export home directories. If you export home directories without using secure RPC, the server and other clients mounting that directory are at risk. An attacker can alter the information stored here or create files, like *an.rhost* files that allows him/her access specifically. With free access the attacker can look for vulnerabilities in the system accounts (*daemon* or *bin*) as much as other users accounts.

n. Do not export a server's executables (such as */bin*, */usr/bin*, ..etc). This can introduce several security risks:

- Allows an intruder to determine what version of executables you are running. This will enable him/her to probe for weak spots easily.
- A flaw on your system configuration may be responsible for the exporting of the binaries on a writable file system. A hacker can modify the binaries, possibly breaking in or cause serious system problems.

To minimize this use diskless client configuration. This means each client maintains a complete set of executables, but stores data subject to change on the central server. Available in some UNIX versions only. If you have to export your executables, do it read-only (*-ro*).

5.2.6 /etc/hosts.lpd

This file is used to allow non equivalent hosts in your network the use of your local printers. It has the same format of the *hosts.equiv* file.

- a. Ensure that the first character of the file is not '!'. Refer to appendix B item 7, [CERT Advisory CA-91:12](#).
- b. Ensure that the permissions on this file are set to 600.
- c. Ensure the owner is set to root.

5.2.7 /etc/ttytab (BSD UNIX systems) /etc/default/login

- a. Ensure that the secure option is removed from all entries that don't need root login capabilities, i.e., all entries with the possible exception of console. It should also be removed from console if you do not want users to be able to reboot in single user mode. If you remove the "secure" option from the console it will force the user to provide a password when booting into single-user mode.
- b. Check that this file is owned by root.
- c. Check that the permissions on this file are 644.

5.2.8 /etc/inetd.conf

This is the internet daemon (inetd) configuration file.

- a. Ensure that the permissions on this file are set to 644.
- b. Ensure that the owner is root.
- c. Disable any services which you do not require, make it a point to find out what each of these entries does. To do this we suggest that you comment out all services by placing a V" at the beginning of each line. Then enable the ones you need by removing the V" from the beginning of the line. In particular, it is best to avoid 'Y' commands and *tftp*, as they have been major sources of insecurities. Other very good candidates are *netstat*, *systat*, and *finger*, they give away too much information about your systems that is very helpful to attackers. For changes to take effect, you need to restart the inetd process. Do this by issuing the following commands:

```
#!/bin/ps -aux | /usr/bin/grep inetd | /usr/bin/grep -v grep  
#/bin/kill -HUP <inetd-PID>
```

5.2.9 Trivial ftp (tftp).

This is similar to ftp but does not require any passwords for authentication; if a host allows tftp without restricting access, usually using the secure option, an intruder can read and write files anywhere in the system. If tftp is not needed, comment it out from the file */etc/inetd.conf* and restart the inetd process. If required use the secure flag to restrict access to a directory that has no valuable information, or you can also run it under the control of a chroot wrapper program.

5.2.10 */etc/services*

This file contains the service names, network port number, protocol name, and a list of aliases. */etc/services* is referenced by client and server programs. a. Ensure that the permissions on this file are set to 644.

b. Ensure that the owner is root.

5.2.11 tcp-wrapper

This is also known as log-tcp. Ensure that you are running this program. This program will allow you to restrict and monitor incoming requests for servers started by the inetd daemon. Refer to Appendix A item 4 on how to obtain tcp_wrapper; customize and install it for your system. Deny all hosts by putting "all:all" in */etc/hosts.deny* and explicitly list trusted hosts who are allowed access to your machine in */etc/hosts.allow*. Wrap all TCP services that you have enabled. In addition to supporting TCP services it also supports UDP-based services.

5.2.12 */etc/aliases*

This file contains the system-wide electronic mail aliases used by sendmail. If an attacker gains access to this file it can create a mail alias that automatically runs a particular program granting him a backdoor to the system.

a. Comment out the "decode" alias by placing a V" at the beginning of the line. For this change to take effect you will need to run */usr/bin/newaliases*:

```
#usr/bin/newaliases
```

If you run NIS (YP), you will then need to rebuild your maps:

```
# cd /var/yp
```

```
#usr/bin/make aliases
```

b. Ensure that all programs executable by an alias are owned by root, have permissions set at 755, and are stored in a systems directory, e.g., */usr/local/bin*.

c. Make sure that no alias runs a program or writes to a file unless you are positively sure what this program does.

5.2.13 Simple Mail Transfer Protocol (SMTP)

SMTP an internet standard used to transfer mail between computer systems. */usr/lib/sendmail* is the UNIX program that ordinarily implements the client and server side of the protocol. Other systems used for this same purpose include *smail*, *MMDF*, and *PMDF*. *Sendmail* is the most common.

a. */etc/sendmail.cf*

The */etc/sendmail.cf* controls sendmail's configuration. Ensure the line starting with "OW" only has an "*" next to it.

b. The main problem that sendmail poses is its complexity. It performs a variety of functions, and it thus requires the permissions necessary to do these functions. Root privileges are required for the following:

- Listen to incoming SMTP connections on port 25
- Act as a particular user in order to read forward and :include: alias files owned by users, run programs specific to those files.
- Execute certain kernel system calls that are restricted to root. An example of this is checking the amount of free space available for incoming messages.
- Protect files in the mail queue from snooping by intruders.

An option to this is to split the functionality of sendmail in two distinct programs. Included in the Trusted Information System firewall kit there is a program called *smap* that has the capability of accepting SMTP connections from outside sites. *smap* receives messages from the Internet and delivers the mail too. *Smmap* and *smmapd* are sendmail wrapper programs. The *smmap* part accepts messages over the net and writes them to a special directory for future delivery. *Smmap* does not require root privileges. All SMTP envelope information is logged by *smmap*. *Smmapd* periodically scans the directory where *smmap* delivers mail. When it finds completed messages, it delivers them to the appropriate user's mail file using sendmail or other program. Using this tool an intruder will never have direct SMTP connection to Sendmail. You can get the TIS Firewall Toolkit from ftp.tis.com.

c. Some examples of Sendmail vulnerabilities follow:

- In its earliest versions sendmail allowed mail to be sent directly to any file on the system, to include */e*
- It supports a "wizard's password", set in a configuration file, that can be used to get a shell on a remote system without actually logging in.
- It allows trusted users, there is a risk of forged mail delivered to the local machines.
- It can be compiled in "debug mode", which in the past has been used to gain unrestricted access to the machine sendmail is running on.
- It used to accept mail with a program as the recipient, allowing remote users to invoke shells and other programs on the destination host.
- It used to do a poor job validating its arguments, allowing users to overwrite arbitrary locations in memory, or provide input that results in very bad things.

d. CERT Advisory CA-96.20 recommends that you install the vendor patches mentioned in the advisory (what version of sendmail you are currently running telnet to it (port 25). If that does not work, there is no easy way of finding out what your version is. You can try downloading a new version yourself. The latest version (sendmail 8.7.6) is available via anonymous FTP from one of the following sites:

<ftp://ftp.sendmail.oro/ucb/src/sendmail/sendmail>

<ftp://nasirc.hq.nasa.gov/toolkits/UNIX/Sendmail-> the latest version here is Sendmail 8.7.5/
8.8.x is Beta

<ftp.cs.berkeley.edu/ucb>

Note: If you don't already run Eric Allman's sendmail.8.6, then it may take you some time to build, install, and configure the system to your needs. Other sendmail configuration files may not be compatible with sendmail 8.6.12.

Check that you have installed the latest patches. This is important as sendmail has been a source of a number of security vulnerabilities. Try the following tests:

```
% telnet <hostname> 25
Wiz
debug
kill
quit
%
```

You should see the response "500 Command unrecognized" after each of the commands 'Wiz', 'debug' and 'kill'. If not, your version of sendmail is vulnerable and should be updated. If telnet is allowed to port 25 a potential intruder can gather the following information:

- The OS, possibly to the version number underdog % telnet mercury.hq.nasa.gov connecting to host mercury.hq.nasa.gov (140.138.123.98, port 25 connection open 220 mercury.hq.nasa.gov Sendmail Sendmail 5.55/mercury ready at Thurs, 7 Nov 96 11:23 EST
expn decode
250 <" | /usr/bin/uudecode">
quit
underdog %
- You can tell if they are running the "decode" alias. The "decode" alias is a major security hole. It allows potential intruders to overwrite any file that is writable by the owner of that alias, that can be any user.

- You can get plenty of information by asking sendmail if an address is acceptable (*vrfy*), or what an address expands to (*expn*). They can also be used to find out if a user is piping mail through any program. It would be a good idea to disable these.

e. Known bugs in Sendmail version 8.7 can be reviewed on:

<ftp://ftp.sendmail.orz/ucb/sre/sendmail/KNOWNBUGS>
<ftp://CS/Berkley.EDU/in/ucb/sendmail/KNOWNBUGS>

f. Increase sendmail logging to a minimum log level of 9. This will help detect attempted exploitation of the sendmail vulnerabilities. To do this, include the following line in the options part of the general configuration information section of the sendmail configuration file:

```
# log level
OL9
```

g. Increase the level of logging provided by syslog. Enable a minimum level of "info" for mail messages to be logged to the console and/or the syslog file. For example, include the following lines in the *syslog.conf* file:

```
mail.info      /dev/console
mail.info      /var/adm/messages
```

Syslog will then need to be instructed to reread the configuration file. This can be done by issuing the command:

```
# /bin/kill -HUP '/bin/cat/ete/syslog.pid'
```

Sample error messages to look for include mail to or from a single pipe ("|"), or mail to or from an obviously invalid user (for example, bounce or blah).

h. Remember that if you are running a frozen configuration file - *sendmail.cf* you will need to rebuild it and restart sendmail before any changes will take effect. To rebuild the frozen configuration file, the command is: `# /usr/lib/sendmail -bz`. To restart sendmail you should kill **all** existing sendmail processes by sending them a TERM signal using kill. Then startup sendmail. Sample commands are:

```
# /usr/bin/ps -auxw | /bin/grep sendmail | /bin/grep -v grep
# /bin/kill <pid> #pid of every running sendmail process
# /usr/lib/sendmail -bd -q1h
```

i. If you want to prevent the exploitation of the sendmail program mailer facility you can use restrict it by using the sendmail restricted shell program (*smrsh*). This shell controls the way that the incoming mail messages can interact with your operating system. When properly

configured it can do things like prevent an attacker from using pipes to execute arbitrary commands on your system.

- j. *mail.local* tool is used to control the way that */bin/mail* program is used on the system.

5.2.14 Majordomo

Majordomo is a program that automates the management of Internet mailing lists. Check that your version is greater than 1.93. You can obtain Majordomo the latest version (1.94) from:

<ftp://ftp.p.gh.net/pub/maiordomo/>

<ftp://ftp.oreatecircle.com/lpub/maiordomo/>

Refer to *CERT Advisory CA-94:11* on majordomo vulnerabilities.

5.2.15 Fingerd

This is the finger server which is started on demand by *inetd*. It supplies information about accounts, including system accounts, on UNIX systems. The information provided by *fingerd* can be used by potential attackers to break into your system. If your version of *fingerd* is older than 5 November 1988, replace it with a newer version.

5.2.16 UUCP

UNIX-to-UNIX copy system. Designed to transfer files (news and email) between UNIX systems and also execute commands on remote systems. UUCP consists of two main programs:

- **uucp**, copies files between machines
- **uux**, executes programs on remote machines

There are three versions of UUCP:

- Version 2
- HoneyDanBer UUCP (HDB/BNU)
- Taylor UUCP

Even though UUCP has many security features built in, the following actions are recommended:

- Disable the uucp account, including the shell that it executes for logging in, if it is not used at your site. If it is used assign a unique password to it.
- Remove any *.rhosts* file at the uucp home directory.
- The *L.cmds* files allows you to specify which commands can be executed by UUCP. *L.cmds* and *uuxqtcmds* are different version of the same file. You should be very careful when adding commands to this file. There are very risky commands, like *rm*, *finger* or *cat*. Check that the file *L.cmds* is owned by root.
- Ensure that no uucp owned files are world writable.

- Check that you have assigned a different uucp login for each system that needs uucp access to your machine.
- Check that you have limited the number of commands that each uucp login can execute to a bare minimum. The *L.cmds* file is used for this purpose.
- If you are running NFS in the same computer that you use for UUCP, the NFS server must not export the UUCP configuration, program, or data directories.
- Configure UUCP so remote systems can retrieve files only from particular directories or you can turn off remote retrieval.
- Request callback. for selected systems to ensure that the UUCP system you are communicating with is not an impostor.
- Make sure that the permissions on */usr/lib/uucp/L.sys* or */usr/lib/uucp/Systems* are 400, readable only by the UUCP user.
- Ensure that no UUCP login has *lusr /spool / uucp / uucppublic* for its home directory.
- Test you mailer to make sure that it will not deliver a file or execute a command that is encapsulated in an address.

5.2.17 WWW Security

The World Wide Web is an effective tool for the exchange and propagation of information. It consists of servers, which make the information available and browser, which are used to access such information and display it on screen.

Security concerns:

- The damage that a malicious client can do to your HTTP server
- The damage that a malicious HTTP server can do to your clients
- Unauthorized individuals getting access to private or confidential documents stored in the server.
- Unauthorized individuals getting access to private or confidential information sent by the remote user being intercepted, for example credit card information.
- Making information about your Web server host available to potential intruders that can help them break into the system.
- Buggy software/scripts that allow outsiders to execute commands on the server that can modify or damage the system.

Secure Web server:

- Clients should never be able to execute arbitrary programs or shell commands on your server.
- CGI scripts running on your server must perform the expected function or return an error message.

General Precautions:

- Run your HTTP Server on a dedicated machine, if possible. Do not mix it with FTP servers.

- Carefully control the external programs your HTTP Server can access.
- Do not run your server as user root. Have it set to run as a nobody user unique to the WWW service.
- Disable automatic directory listings.
- Be aware of all the configuration options for the server you are using, and that those options are set appropriately.
- Limit or prohibit server-side includes.
- Limit the number of logins available. Remove the accounts that have been inactive or are unnecessary.
- Do not NFS mount or export any directories.
- Do not run a mail server.
- Passwords should be chosen properly and carefully. Run the Crack program against them to detect poorly-chosen ones. Crack can be obtained from:
ftp://ftp.cert.org/pub/tools/crack
- Any service you are not using, turn off. This services can include but are not limited to ftp, sendmail, NIS, gopher, NFS, finger, etc Comment the services out of the */etc/inetd.conf* file.
- All shells and command interpreters you do not need, remove.
- Permissions on system files should be set correctly. Use COPS to make sure permissions are set correctly. You can get COPS from the following site:
ftp://ftp.cert.org/pub/tools/cops/
- Make sure you are using all appropriate logging files and make sure you check them for suspicious activity. Use Tripwire to scan system logs. Can be obtained from:
ftp://coast.cs.purdue.edu/pub/COAST/Tripwire
- Limit access to the document root directory, where HTML documents are stored, and server root directory, where log and configuration files are kept, to trusted Web authors and an official Web Administrator.
- Prevent general access to the server log files.

Tips:

Permissions on your Server's Directory Structure

- NCSA (we are using it as an example) server has six directories:
 - *cgt-bin* - Holds CGI scripts
 - *conf* - Holds server configuration files
 - *htdocs* - Holds Web documents
 - *icons* - Holds Web documents
 - *logs* - Records server activity
 - *support* - Holds supplemental programs for the server. The best approach is to setup the Web server directory with root ownership:

drwx-x-x	8	root	www	1024	Nov 23 09:45	<i>cgi-bin/</i>
drwx	2	root	www	1024	Nov 26 10:50	<i>conf/</i>

drwr-xr-x	2	root	www	1024	Nov 23 09:45	<i>htdocs/</i>
-rwx	1	root	www	482168	Aug 6 09:45	<i>httpd*</i>
drwxrwxr-x	2	root	www	1024	Dec 1 18:15	<i>icons/</i>
drwx—2	2	root	www	1024	Nov 26 10:50	<i>logs/</i>
drwr-xr-x2	2	root	www	1024	Nov 23 09:45	<i>support/</i>

cgi-bin directory has permissions of 711, which allows the *httpd* server to run the programs it contains, but does not allow a person on the server to view the contents of the directory.

- Configuration Files

The following files exist under the *conf* directory, in the NCSA server:

File	Purpose
<i>access.conf</i>	Controls access to the server's files.
<i>httpd.conf</i>	Configuration file for the server.
<i>mime.types</i>	Determines the mapping of file extensions to MIME file types.
<i>Srm.conf</i>	Server Resource Map. Contains additional server configuration. Determines the document root, whether or not users can have their own Web directories, the icon for directory listings, the location of the CGI script directory, document redirections, and error messages.

These files should only be read, write, and accessible to root:

```
-rw--    1 root  riley  954   Aug  6  01:00 access.conf
-rw--    1 root  riley 2840   Aug  6  01:05 httpd.conf
-rw--    1 root  riley 3290   Aug  6  00:40 mime.types
-rw--    1 root  riley 4280  Nov 18  10:00 srm.conf
```

The reason for this is that those files can be used by an unauthorized user to destroy the server or the entire system.

- Eavesdropping

The fact that sensitive information is transmitted over the network is a risk that has to be considered when setting up a Web Server. There are two ways that potential attackers can eavesdrop, over the wire, or looking at the log files in the web server. To prevent this you either must have a physically secure network or some form of encryption for your information.

* Four possible ways of encryption are:

1. Link Encryption - encrypting the phone line.

2. Document encryption - encrypting the documents, an example is PGP.
3. SSL (Secure Socket layer) - System designed by Netscape that provides encrypted TCP/IP pathway between two hosts on the Internet. Encrypts any TCP/IP protocol. More information on:
<http://home.netscape.com/newsref/std/SSL.html>
4. SHTTP (Secure HTTP) - Encryption system for HTTP, designed by Commerce Net. More information on: <http://www.eit.com/projects/s-http/>

* Log Files are created by most Web servers. These record a vast amount of information about requests being submitted to the server. Examples of these files and the information they contain follows. We will use the NCSA httpd server as a model:

1. *access - log* - Contains a list of the people that access the server.
2. *agent_log* - Contains a list of the programs that have been used to access the server.
3. *error - log* - Contains a list of errors experienced by the server. The can be generated by CGI scripts or the server itself
4. *refer-log* - Contains entries that include URL the browser previously visited and the URL is currently viewing.

5.3 Ftpd & Anonymous Ftp

5.3.1 Versions

a. Ensure you are using the most recent version of the ftp daemon.

b. Consider installing the Washington University ftpd if you don't already have it. This can log all events and provide users with a login banner. Do not install any versions prior to *wu-ftp 2.4*. Refer to C.7, [CERT Advisory CA-94.07](#). It is available via anonymous ftp; warning: versions of *wu-ftp* prior to 2.4 are extremely insecure and in some cases have been trojaned. For BSDI systems, patch 005 should be applied to version 1. 1 of the BSD/386 software. It is available via anonymous ftp from:

ftp://nasirc.hq.nasa.gov/toolhits/UNIX/Wuftp/*

<ftp://ftp.bsdi.com/Ibsdi/patches/README>

<ftp://ftp.bsdi.com/bsdi/patches/?UI10-005>

(? will be B or S for the Binary or Source version)

5.3.2 SITE EXEC

a. Check to make sure your ftp server does not have the *SITE EXEC* command. Do this by telneting to port 21 and typing *SITE EXEC*. If your ftp daemon has *SITE EXEC* make sure you have the most recent version of the daemon (eg, *wu-ftp 2.4*). In older versions of ftpd *SITE EXEC* allows anyone to gain shell via port 21.

b. Check that any commands from *~ftp/bin*, *~ftp/usr/bin*, *~ftp/sbin* or similar directory configurations that can be executed by *SITE EXEC* do not contain system commands or include a

shell (e.g., `~ftp/bin -> /bin`). If they do contain system commands it is possible for local users to gain root access. Additional information is available via anonymous ftp from:

<ftp://ftp.quscert.org.au/pub/auscert/auscert-advisory/AA-94.01.ftp.Configuration.Advice>
<ftp://ftp.auscert.org.au/pub/auscert/auscert-advisory/AA-9504.wu-ftpd.misconfimuration.vulnrabil>
[ity](#)

5.3.3 Configuration of Your Anonymous Ftp Server

a. Setting up the anonymous FTP directories

The root directory and its subdirectories, under anonymous ftp, SHOULD NOT be owned by the FTP ACCOUNT OR BE IN THE SAME GROUP AS THE FTP ACCOUNT. The reason for this is that if any of these directories are owned by ftp or are in the same group as the ftp account and are not write protected, an intruder will be able to add or modify files (example, the rhosts file). A good strategy is to have these directories owned by root. This way the ftp directory and its subdirectories will be part of the system group, and will be protected so only root has write permissions, adding security to the setup. An example of the setup follows:

drwxr-xr-x	7	root	system	512	Mar 1	15:17./
drwxr-xr-x	25	root	system	512	Jan 4	11:35../
drwxr-xr-x	2	root	system	512	Dec 21	14:43 bin/
drwxr-xr-x	2	root	system	512	Mar 12	15:33 etc/
drwxr-xr-x	10	root	system	512	Jun 5	10:54 pub/

Files and libraries, especially those used by the FTP daemon and the ones in the `-ftplbin` and `-ftpletc` directories, should have the same protections as the parent directories.

b. Using proper password and group files

It is strongly advised that Codes not use the system's `letclpasswd` or `letclgroup` as the password and group files for the `-ftp/letc` directory. These files should be owned by root. Dummy versions of both, the `-ftp/letc/1passwd` and `-ftp/letc/1group` should be implemented in Headquarters ftp servers. These files should only include those entries relevant to the FTP hierarchy or needed to show owner and group names. Please make sure there are no entries in the password field. An example of a password file and a group file from the anonymous FTP directory on <ftp.hq.nasa.gov>:

~ftp/etc/passwd:

```
cpwg:*:2101:20:TCP-IP Working Group::  
tools:*: 2102:20:UNIX Tools::  
ftp:*:17:90:Anonymous FTP::
```

~ftp/etc/group:

```
hqlocal:*:20:  
ftp:*:90:
```

Check that you do not have a copy of your real `/etc/passwd` file as `~ftp/etc/passwd`. Create one from scratch with permissions 444, owned by root. It should not contain the names of any accounts in your real password file. It should contain only root and ftp. These should be dummy entries with disabled passwords e.g.: **root:*:O:O:Ftp maintainer::ftp:*:400:400:Anonymous ftp::**

Check that you do not have a copy of your real `/etc/group` file as `~ftp/etc/group`. Create one from scratch with permissions 444, owned by root.

c. Providing writable directories in the anonymous FTP configuration

Operating an anonymous FTP service that allows users to store files involves a risk. If you are going to provide this functionality to the users you have to do it in a secure way. The possibility for abuse of copyrighted software or to trade information on compromised accounts and password files is extremely high. The possibility also exists for files systems to be maliciously filled causing denial of service problems. There are several ways to address this problem that we will discuss. The first method is to use a modified FTP daemon. The second is to provide restricted write capability through the use of special directories. The third involves the use of a separate directory.

- Modified FTP daemon

If the FTP server is to offer a “drop off” service, then the server should be implemented with a modified FTP daemon that will control access to the “drop off” directory. This is the

best way to prevent the unwanted use of writable areas. The following modifications are recommended:

1. Implement a procedure where any file dropped off cannot be accessed until the system administrator examines the file and moves it to public directory.
2. Limit the amount of data transferred in one session.
3. Limit the overall amount of data transferred based on available disk space.
4. Increase logging to enable earlier detection of abuses.

If the Code is interested in modifying the FTP daemon, the source code is available from the UNIX vendor. Public domain sources are available from the following:

wuarchive.wustl.edu `~ftp/packages/wuarchive-ftp`

ftp.uu.net `:~ftp1systems1unix1bsd
sources/libexec/ftpd`

gatekeeper.dee.com `~ftp/ub/DEC/gwtools/ftpd.tar.Z`

NASA Headquarters has reviewed and evaluated both the **wuarchive-ftpd** and the **uunet ftpd**. Currently the **wu-ftpd** program provides the greatest level of security and auditing deemed necessary to support a secure anonymous FTP server at NASA Headquarters and is therefore the "preferred" daemon.

- Using protected directories

If you are planning to offer a "drop off" service and are unable to modify the FTP daemon, it is possible to control access by using a maze of protected directories. This method requires prior coordination and cannot guarantee protection from unwanted use of the writable FTP hierarchy, but has been used effectively on many hosts.

Protect the top level directory (`~ftp/incoming`) giving only execute permission to the anonymous user (`chmod 751 ~ftp/incoming`). This will permit the anonymous user to change directory (`cd`), but will not allow the user to view the contents of the directory.

```
drwxr-x-x 4 root system 512 Jun 11 13:29 incoming/
```

Create subdirectories in the `~ftp/incoming` using names known only between your local users and the anonymous users that you want to have "drop off" permission. The same care used in selecting passwords should be taken in selecting these subdirectory names because the object is to choose names that cannot be easily guessed. DO NOT use the example directory names of `e4UhtM6q` and `Lv9Wur2x`.

drwxr-x-wx	10	root	system	512	Jun 11	13:54	<i>e4UhtM6q/</i>
drwxr-x-wx	10	root	system	512	Jun 11	13:54	<i>Lv9Wur2x/</i>

This will prevent the casual anonymous FTP user from writing files in the anonymous FTP file system. It is important to realize that this method does not protect an implementation against the result of intentional or accidental disclosure of the directory names. Once a directory name becomes public knowledge, this method provides no protection at all from unwanted use of the area. Should a name become public, the administrator may choose to either remove or rename the writable directory.

*** Using a single disk drive**

If the implementation will offer a "drop off" service and is unable to modify the FTP daemon, it may be desirable to limit the amount of data transferred to a single file system mounted as *~ftp/incoming*. If possible, dedicate a disk drive and mount it as *~ftp/incoming*. The system administrator should monitor this directory (*~ftp/incoming*) on a continuing basis to ensure that it is not being misused.

*** Permissions**

1. Ensure no files or directories are owned by the ftp account or have the same group as the ftp account. If they are, it may be possible for an intruder to replace them with a Trojan version.

2. Ensure that the anonymous ftp user cannot create files or directories in any directory.

3. Ensure that the ftp user can only read information in public areas.

4. Ensure that the permissions of the ftp home directory (*~ftp/*) are set to 555 (read, no write, execute), owner set to root (not ftp).

5. Ensure that the system subdirectories *~ftp/etc* and *~ftp/bin* have the permissions 111 only; owner set to root.

6. Ensure that the permissions of files in *~ftp/bin/** have the permissions 111 only, owner set to root.

7. Ensure that the permissions of files in *~ftp/etc/** are set to 444, owner set to root.

8. Ensure */usr/spool/mail/ftp* is owned by root with permissions 400. If you need email for the ftp admin (e.g., for problem reporting) use an alias.

5.4 Password & Account Security

5.4.1 Proactive Checking

a. Use *npasswd*, *anlpasswd*, or *passwd+* to proactively screen passwords as they are entered. These programs run a series of checks on passwords when they are set and can help to screen out poor passwords. They were not designed to work with shadow password systems. Refer to Appendix A item 3 for how to obtain *npasswd* and *passwd+*. *Anlpasswd* can be found at info.mcs.anl.gov.

b. Check passwords periodically with *Crack*. *Crack* is a program that scans the */etc/passwd* file for easy to guess passwords. Refer to Appendix A item 1 for how to obtain *Crack*.

c. Apply password aging (if possible). Password aging is a "lifetime" set for passwords by the system administrator. It forces password expiration at set times. Setting this between one and six months is acceptable.

5.4.2 Root Password

Restrict the number of people who know the root password. These should be the same users registered with groupid 0 (e.g., wheel group on SunOS). Typically, this is limited to at most 3 or 4 people.

5.4.3 NIS and */etc/passwd* Entries

a. Do not run NIS if you do not really need it. Check that the only machines that have aY entry in the */etc/passwd* files are NIS (YP) clients; i.e. not the NIS master server. There appears to be conflicting documentation and implementations regarding theY entry format and so a generic solution is not available here. It would be best to consult your vendor's documentation. Some of the available documentation suggests placing a " in the password field, which is not consistent across all implementations of NIS. We recommend testing your systems on a case- by-case basis to see if they correctly implement the" in the password field. To do this, follow the steps below:

Try using NIS with the "*" in the password field for example; +*:O:O::: If NIS users cannot log in to that machine, remove the" and try the next test.

With the "*" removed, try logging in again. If NIS users can log in and you can also log in unauthenticated as the user '+', you have a problem. Your vendor needs to change their version of NIS. If NIS users can log in and you cannot log in as the user '+', your implementation should not be vulnerable to this problem.

b. Check that */etc/rc.local* is set up to start *ybind* with the -s option. This may not be applicable on all systems. Check your documentation.

5.4.4 Password Shadowing and C2 Security

a. Implement C2 security controls if available for your system. UNIX systems that offers at least so-called C2-level security features have shadow password files, or the capability to install them. Consult your vendor documentation for details.

b. Implement vendor supplied password shadowing or a third party product if you cannot run full C2 security. Password shadowing prevents users from reading each others encrypted passwords. If you use shadow password files, make sure that there are no backup copies of the shadow password file that are publicly readable on the system.

c. Periodically audit your password and shadow password files for unauthorized additions or inconsistencies.

5.4.5 Administration

a. Ensure that you regularly audit your system for dormant accounts and disable any that have not been used for a specified period, say 3 months. Send outaccount renewal notices and delete or disable substitute an * for the encrypted password any accounts of users that do not reply.

b. Check that all accounts have passwords.

c. Ensure that any user area is adequately backed up and archived.

d. Regularly monitor logs for successful and unsuccessful su attempts. Some SystemV-based UNIX versions can determine logging settings via */etc/default/su* file.

e. Regularly check for repeated login failures. SystemV-based UNIX versions (Solaris included) */var/adm/loginlog*.

f. Regularly check for LOGIN REFUSED messages.

5.4.6 Special Accounts

a. Check that there are no shared accounts other than root; i.e., more than one person should not know the password to an account. Use the group ID mechanism to set up this scenario.

b. Disable guest or default accounts. Better yet, do not create guest or default accounts.

c. Use special groups (such as the "wheel" group under SunOS) to restrict which users have access to root.

d. Change or disable all default vendor accounts shipped with the Operating System. This should be checked after each upgrade or installation. An example of this are uucp and daemon.

e. Disable accounts that have no password which execute a command, for example "*sync*" *
Preferably, remove these accounts entirely. Check that they do not own any files before you do so.

5.4.7 Root Account

- a. Make sure root does not have a *~/.rhosts* file.
- b. Do not login directly as root, login into your own account and use the *su* command.
- c. Make sure "." is not in root's search path.
- d. Make sure root's login files do not source any other files not owned by root, or which are group or world writable.
- e. Make sure *root cron* job files do not source any other files not owned by root or which are group or world writable.
- f. Use absolute path names when root, i.e., */bin/su*, */bin/find*, */bin/passwd*. This is to stop the possibility of root accidentally executing a Trojan horse. To execute commands in the current directory, root should prefix the command with ".", e.g., */command*.

5.5 File System Security

5.5.1 General

a. Check that there are no *exrc* files on your system that have no legitimate purpose. These may inadvertently perform commands that may compromise the security of your system if you happen to start either *vi* or *ex* in a directory which contains such a file. To find *exrc* files:

```
#/bin/find/\(-fstype 4.2-o -prune\) -name '.exrc'\ -exec /bin/cat {} \;-print.
```

b. Check that any forward files in user home directories do not execute a command or run a program. The mailer maybe fooled into allowing a normal user privileged access. The following command will locate and print forward files:

```
#bin/find / \(-fstype 4.2 -o -prune\) \-name '.forward'-exec /bin/cat {} \;-print.
```

5.5.2 /etc/rc.local

a. Check */etc/rc.local* does not *chmod 666 motd*. This allows users to change system message for the day.

b. Ensure that the line `"rm -f/tmp/t 1"` (or similar) exists in `/etc/rc.local` to clean up the temporary file used to create `/etc/motd`. This should occur before the code to startup the local daemons.

5.5.3 /usr/lib/expreserve

a. Replace versions of `/usr/lib/expreserve` prior to July 1993 with a recommended patch from your vendor. If this is not possible, then remove execute permission on `lusr/lib/expreserve`:

```
#/bin/chmod 400 /usr/lib/expreserve
```

This will mean that users who edit their files with either `vi` or `ex` and have their sessions interrupted, will not be able to recover their lost work. If you implement the above work around, please advise your users to regularly save their editing sessions.

5.5.4 External File Systems/Devices

a. Mount file systems non-setuid and read-only where practical; refer to NFS.

5.5.5 File Permissions

a. Check that the permissions of `/etc/utmp` are set to 644.

b. Check that the permissions of `/etc/sm` and `/etc/sm.bak` are set to 2755.

Check that the permissions of `/etc/state` are set to 644.

d. Check that the permissions of `/etc/motd` and `/etc/mtab` are set to 644.

. Check that the permissions of `/etc/syslog.pid` are set to 644.

f. Remove setgid privileges on `/usr/kvm/crash`. A group of `kmem` allows to read the virtual memory of a running system. (`# /bin/chmod g-s/usr/kvm/crash`).

g. Consider removing read access to files that users do not need to access.

h. Ensure that the kernel (eg. `/vmunix`) is owned by root, has group set to 0 (wheel on SunOS) and permissions set to 644.

L Ensure that `/etc`, `/bin`, `/usr/etc`, `/usr/bin` and `/tmp` are owned by root and that the sticky-bit is set on `/tmp`, i.e., permissions on `/tmp` should be: `drwxrwxrwt`. You should implement COPS or Tiger to check for this. Refer to Appendix A item 2 for information on where to obtain these.

j. Ensure that there are no unexpected world writable files or directories on your system. The following commands find world directories.

```
#/bin/find /-type f -perm -22 -exec ls -l {} \;  
#/bin/find /-type d -perm -22 -exec ls -ld {} \.
```

- k. Check that files which have the SUID or SGID bit enabled, should have it enabled:

```
#/bin/find /-type f \(-perm -004000 -o -perm -002000\) \exec ls -l{} \;
```

- l. Check the umask value for each user and ensure it is set to something sensible like 027 or 077. Refer to Appendix C item 1 for a shell script to check this.
- rn. Root's umask should be 057 or more restrictive. This can be modified by hand by the system administrator when required.

5.5.6 Files Run By Root

- a. It is recommended that anything run by root, should be owned by root, should not be world or group writable and should be located in a directory where every directory in the path is owned by root and is not group or world writable.

- b. CHECK the contents of the files listed below for the root account. Any programs or scripts referenced in these files should meet the above requirements:

- ~/login, ~/.profile and similar login initialization files
- ~/exrc and similar program initialization files
- ~/logout and similar session cleanup files
- crontab and at entries
- /etc/rc* and similar system startup files

- c. If any programs or scripts referenced in these files source further programs or scripts they also need to be verified.

5.5.7 Bin Ownership

Many systems ship files and directories owned by bin. This varies from system to system and may have serious security implications. Change all non-setuid files and all non-setgid files and directories that are world readable but not world or group writable and that are owned by bin to ownership of root, with group id 0 (wheel group under SunOS). Anything else should be verified with the vendor.

5.5.8 Tiger/COPS

- a. Run one or both of these. Many of the checks in this section can be automated by using these programs. For information on where to get these programs see Appendix A item 2.

5.6 SunOS Security

The following is a list of helpful security checks that relate specifically to SunOS 4.1.x. This is not necessarily a complete list.

5.6.1 IP Forwarding

a. Check that IP forwarding is disabled. You will need the following line in the kernel configuration file:

options'IPFORWARDING=-1. For information on how to customize a kernel, see the file:

/usr/sys/arch/conf/README.

5.6.2 Framebuffers /dev/fb

If somebody can log in to your Sun workstation from a remote source, they can read the contents of your Framebuffer, which is */dev/fb*. Sun provides a mechanism which allows the user logging in to have exclusive access to the Framebuffer, by using the file */etc/fstab*. A sample */etc/fstab* file:

```
#
# File:                /etc/fstab

# Purpose:             Specifies that upon login to /dev/console, the owner, group and
                       permissions of all supported devices, including the framebuffer, will be
                       set to the user's username, the user's group and 0600.

# Comments:           SunOS specific.

# Note:               You cannot use \ to continue a line.

# Format:
# Device              Permission   Colon separated device list.
#
/dev/console          0600          /dev/fb
/dev/console          0600          /dev/bwoneO:/dev/bwtwoO
/dev /console         0600          /dev /cgoneO. /dev /cgtwoO: /dev/cgthreeO
/dev/console         0600          /dev /cgfourO: /dev /cgsTXO: /dev/cgeightO
/dev/console         0600          /dev /cgnlneO: /dev /cgtweiveO
#
/dev/console         0600          /dev /kb: /dev /mouse
/dev/console         0600          /dev/fdOc:/dev/rfdOc
```

After the above file has been created, reboot your machine, or log out fully, then log back in again. Read the man page for *fstab(5)* for more information.

5.6.3 /usr/kvm/sys/*

Check all files and directories under */usr/kvm/sys/* are not writable by group. In SunOs 4.1.4 the default mode is 2775 with group staff, allowing users in group staff to Trojan the kernel.

5.6.4 /dev/nit (Network Interface Tap)

- a. Run the CERT tool *cpm* to check if your system is running in promiscuous mode.
- b. Disable the */dev/nit* interface if you do not need to run in promiscuous mode. For SunOS 4.x and Solbourne systems, the promiscuous interface to the network can be eliminated by removing the */dev/nit* capability from the kernel. Once the procedure is complete, you may remove the device file */dev/nit* since it is no longer functional.
- c. Apply "method 1" as outlined in the System and Network Administration manual, in the section, "Sun System Administration Procedures," Chapter 9, "Reconfiguring the System Kernel." Excerpts from the method are reproduced below:

```
# cd /usr/kvm/sys/sun[3,3x, 4,4c]/conf
# cp CONFIG - FILE SYS - NAME
```

[Note that at this step, you should replace the CONFIG_FILE with your system specific configuration file if one exists.]

```
# chmod +w SYS_NAME
# vi SYS_NAME
#
# The following are for streams NIT support. NIT is used by # etherfind, traffic,
rarpd, and #ndbootd. As a rule of thumb, # NIT is almost always needed on a server
and almost never # #needed on a diskless client.
#
pseudo-device    snit    streams NIT
pseudo-device    pf      # packet filter
pseudo-device    nbuf    # NIT buffering module
```

[Comment out the 3 "pseudo-device" lines; save and exit the editor before proceeding.]

```
# config SYS_NAME
# cd..ISYS NAME
# make
# mv /vmunix /vmunix.old
# cp vmunix/vmunix
#/etc/halt
> b
```

This step will reboot the system with the new kernel. Note that even after the new kernel is installed, you need to take care to ensure that the previous `vmunix.old`, or other kernel, is not used to reboot the system. See CERT Advisory CA 94.01. SUN patches can be obtained from:

<ftp://www-nasirc.nasa.gov/toolkits/>

5.6.5 System Initialization

a. Determine if the UNIX system software is the most current version. The software version can be obtained through the `uname -a` command

b. Determine that a documented inventory of system software is available and all vendor supplied software is supported.

c. Verify that a shadow password file is being used.

`/etc/shadow`

d. Confirm activation of the audit facility through the existence of `/etc/security/audit-control` and the existence of the `auditd` daemon being started in the `/etc/rc.local`. Review the audit parameters to ensure that they are set up properly.

e. Check the `umask` value to ensure it is set as '027'.

f. Examine the rc scripts that the system executes when it is booted as well as the file attributes of those scripts.

g. Review the `/etc/nsswitch.conf` file and ensure that a logical configuration has been set up for NIS or NIS+. Also if NIS or NIS+ is being used verify on a sample basis that the client machines and the server have the correct domain name has been defined in `/etc/default` domain.

h. Verify that users other than root do not have Write access to system directories. Use the command:

```
find / -type d -exec ls -ld {} \;
```

Pay special attention to the following directories:

- `*/`
- `*/bin`
- `*/usr`
- `*/usr/etc`
- `*/etc`

**/dev*
**/usr/lib*
**/export*
**/usr/sccs*
**/home*
**/Ivar*
**/kernel*
**/var/admin*
**/lib*
**/var/crash*
**/opt*
**/var/spool/cron*
**/sbin*
**/ var /spool /crontabs*
**/Sys*
**/Vol*
**/tmp*

i. Determine if access is restricted to the **crontab**, and the at commands. This can be accomplished by either listing users permitted to use the command in the file */ var /spool /cron /cron. allow* and the */ var /spool /cron lat. allow* or in the list of user not permitted to access the command in the file */ var /spool /cron /cron. deny*. Note: if both files exist the *cron.deny* is ignored.

j. Verify that the root owned crontabs are only accessible by root.

/var/adm/cron/crontab

k. Verify that the automatic startup command scripts files are properly protected and that only valid entries are included within these files:

/etc/inittab
/etc/init.d
/sbin/rc#.d (where # equals run level)
/etc/rc#.d (linked to */sbin/rc#.d*)

1. Determine that system "C" compilers are not located on the production machines.

5.6.6 System Administration

a. Obtain a listing of all the user accounts

/etc /passwd or NIS+ *passwd*

Determine that all users are authorized to sign on to the system

b. Obtain a listing of all group accounts

/etc/group or NIS+ *group table*

Determine that all users within a group are properly authorized.

c. Determine that proper administration procedures are in effect.

- Training
- Procedures
- Change Control

d. Using the following command to determine if any uids are being shared:

```
awk -f. '{if (priv == $3) print, priv = $3}'
```

e. Determine that Vendor access is properly restricted. Review the */etc/passwd* or *passwd.domain_name* (where NIS+ is being used) for vendor ids. Verify that these ids are only enabled when needed and revoked after use.

f. Vendor supplied users and group ids.

Determine that most vendor supplied uids are set up to prevent login capability.

- * root Restrict (if possible)
- * daemon Restrict
- * bin Restrict
- * sys Restrict
- * uucp Restrict (if possible)
- * noaccess Restrict
- * nobody Restrict
- * adin Restrict if possible
- * lp Restrict
- * listen Restrict
- * nuucp Restrict if possible
- * smtp Restrict

Restriction can be accomplished by placing an “NP” in the password field.

g. Determine that the default profile for the system has the proper parameters

/etc/default/login

PATH settings determine if the path parameter is valid and not writable by unauthorized users.

umask = 027

h. Determine that the assignments of home directories is properly established.

Scan the passwd file to see if users are assigned to a proper home directory.

Review the */etc/auto.home* or *auto - home-domain-name* to ensure that the proper home directories are being assigned.

i. Ensure that root and system directories are properly protected and that only root has access to these directories.

j. Password Controls Verify that password rules are being followed and meet the minimum characteristics.

- Password lengths */etc/default/passwd*
- Password alpha/numeric - min/max
- Password aging
- Password attempts before lock out
- Password dictionary check
- Emergency passwords

Verify that the system terminates a logon attempt after three invalid passwords and the system disables the ID after six invalid logon attempts. Because this protection is not offered by SUN system software, external security packages may be used for disabling of ids after successive failed access attempts.

Note: Invalid successive login attempts of five or more will be logged to */usr/adm/messages* if the following line is uncommented from */etc/syslog.conf*

auth.notice/usr/adm/messages

k. Root Signon

Verify that root is restricted to certain devices for signon. Or that root can not directly signon but must su to the root account.

Review */etc/default/su*

To deny direct login of root on all devices, the following line should be commented:

CONSOLE=/dev/console

l. su logging

Verify su activity is logged and reviewed (especially su's to root). su activity will be logged if the following entries are present in /etc/default/su

SULOG=/var/adm /messages SYSLOG=YES

m. Adopted Programs

Ensure that all root owned setuid/setgid are authentic.

n. Determine if the chroot call is being used. Sun provides a chroot front end program /usr/ete/chroot which executes the system call to restrict user's access. This call makes the directory specified appear as the root directory (/) to the user.

o. Determine if the system administrators are restricting the user functionality through menu scripts which are called from the user's .profile files. These menus and submenus should have trap commands called at the beginning of the profile file to ensure CTRL-C and other keyboard interrupts are ignored.

p. Determine that all accounts have passwords.

For C2 configured systems, all users should have an "NP" in the password field of the /etc/shadow

q. Determine if the following command files are properly protected:

- * /bin owned by root with 755 permissions
- * /usr/bin
- * /
- * /lib
- * /usr/etc
- * /usr
- * /usr/lib
- * /etc
- * /usr/ucb

r. Login Files

Files such as */etc/logindevperm* and */etc/ttydefs* control various aspects of the login process. Write access to these files should be restricted to an administrative account. Obtain a copy and review a listing of */etc/ttydefs*.

All ports should have an initial program of */usr/bin/login* or similar

s. Crash Dump Files

In the event of a system crash, the contents of main memory are written to the file */var/adm/crash.vmcore.n* (n is incremented with each instance).

The kernel image is written to */var/adm/crash.vmunix.n*. The data in these files may be sensitive and should be appropriately restricted to root or adm accounts.

t. Backups

Ensure that system and application backups are taken on a regularly and consistent basis.

5.6.7 Network Security

Obtain a copy of the network diagram and compare it to the actual network structure.

a. Determine that only valid entries are included in the trusted host file:

\$cat hosts.equiv

b. Verify that any admintool being used is properly secured by using the secure RPC. This can be verified by listing the contents of */etc/inet/inet.conf* which should look like:

/usr/sbin/inetd.conf admind -S2

c. Are password encrypted across the network? Or is a Challenge and Response system being used such as:
Kerberos and/or IBM's DCE.

d. Ensure that the following network files are properly protected:

/etc/exports Validate the entries and do not use root= option if possible

Verify that the **-access** and the **-secure** options are present on all filesystems named:

- * */etc/hosts*
- * */etc/services*
- * */etc/protocols*
- * */etc/inetd.conf* /*etc/inet/inetd.conf* contains the active services of the network
- * */etc/netgroup* restrict to applications
- * */etc/hosts.equiv* Validate that the entries are local hosts
- * */etc/ethers*
- * */home/.rhost* remove any of these files or set the rlogin daemon (rlogind) has the **-1** flag set.

If NIS+ is being used, the listing can be obtained with the following command:

```
niscat -h table. domain_name
```

e. UUCP

Determine if uucp is being used. Ensure that the uucp account is password controlled and that a separate uucp account is established for each machine that requires access.

Review the */etc/uucp/Permission* file for valid entries

Review the */etc/uucp/Systems* file for valid entries

f. FTP

Verify that the File Transfer Protocol (FTP) is secured and that no Anonymous FTP accounts exist. If there is a business reason for the use of Anonymous FTP, ensure the following:

- * An asterisk is present in the password field for account, FTP
- * A home directory (e.g., */usr/ftp*) has been defined for FTP.
- * */Bin* and */etc* subdirectories under this directory should be created. The home directory should be owned by FTP while the */bin* and */etc* subdirectories should be owned by root.
- * A public subdirectory should also be created under the dome directory for placement
- * and retrieval of transferred files. FTP should only have write access to the public subdirectory and if copies of password files are placed in the */etc* subdirectory, the only account listed should be that of FTP.

5.6.8 User Security

a. Using a complete listing or a sample of the user accounts, determine if their home directory is protected from unauthorized access

Permissions = rwxr-x-

b. Ownership

Determine that the files within the user's directory are owned by the user.

c. Shell

Determine if the user has a valid entry point within the system.

Most users should be placed directly into an application by executing the application from the user's profile or from the entry in the */etc/passwd* file or as a start point for the user with the execution of a startup program.

d. Critical Files

Review the user's home directory for the following files:

- * *.rhost*
- * *.netrc*

These files contain data for logging into remote hosts for file transfer and remote access.

Review all *.exrc* files to ensure that they are not in world-writable directories such as */tmp*. These files should be located only in home directories.

e. User Profile

- * *.profile*
- * *.login*
- * *.cshrc*
- * *.kshrc*
- * *.logout*

Ensure that the *.profile* is properly protected and does not contain startup parameters that conflict with the system startup parameters.

f. User Training

Determine that there is a process in place to educate users as to appropriate security procedures.

g. User Security Notification

Determine that the user is notified of the date and time of their last login.

5.6.9 Audit Files

- * Audit files/parameters reside in */etc/security/* and log files reside in */etc /security /audit-control*
- * Additional logging information is defined in */etc/inetd.conf*
- * System activity is logged in */var/adm* directory
- * The audit id and root should be the only accounts with access to log files and parameters.

5.7 IRIX Security

The following is a list of security issues that relate specifically to the IRIX operating system. This is not necessarily a complete list.

5.7.1 /usr/lib/vadmin/serial_ports

The `/usr/lib/vadmin/serial_ports` program is used to initialize the data files for the serial ports on your system. This only be applies to IRIX Version 4 systems, or Version 5 systems that still contain the serial-ports program. Disable this program:

- a. If you are not using the serial ports on your IRIX Version 4 system.
- b. If you are using serial ports and do not wish to change the configuration of those ports.
- c. If you are using version 5 and the serial-ports program is present. This program has been superseded by `/usr/Cadmin./bin/cports` on Version 5 and, therefore, is no longer required. `/usr/lib/vadmin/serial_ports` can be disabled by the command:

```
#/bin/chmod 700 /usr/lib/vadmin/serial-ports
```

If you intend changing the serial port configuration, you can still disable the `serial_ports` program as above. To change the serial port configuration, run the `serial_ports` program as root.

5.7.2 IRIX Patches

Some patches are available via anonymous ftp from:

<ftp://nasirc.hq.nasa.gov/toolkits/SGI Patches/>*

5.7.3 IRIX Configuration Issues

These issues apply to any version of IRIX:

Some accounts come with no passwords, including `guest`, `4Dgifts`, `demos`, `tutor`, and most importantly `lp`. Examine your `/etc/passwd` file and lock all unnecessary open accounts. The `guest` account is used by some IRIX parts and remote print clients need access to `lp`, so this is a situation you need to study closely. Please refer to CERT-95:15, which can be obtained from:

<ftp://info.cert.org/pub/cert advisories>

Also take a look at *SGI Advisory 19951002-01-1*, which can be obtained from:

<ftp://sgiaate.sai.com/Security/>

`"xdm"` does `"xhost +"` by default when login in. What this means is that anyone is allowed to open windows on your display and even record what you type at your keyboard. To close this remove the `"xhost +"` from `/usr/lib/X11/xdm/Xsession`, `/usr/lib/X11/xdm/Xsession-remote` and, in IRIX 5.x, `/usr/lib/X11/xdm/Xsession.dt`. To correct this vulnerability follow this instructions: 1) Become root.

% **/bin/su -**

Password:

#

2) Edit the file */usr/lib/X11/xdm/Xservers* and add the line below. Normally there is only 1 line, but for TKO, be sure this is added for each Xserver.

add option '-shmnumclients 0'

3) Save file.

4) Edit the */usr/lib/X11/xdm/xdm-config* and make the following change.

DisplayManager*authorize: off

to

DisplayManager*authorize: on

5) Save the file.

6) Edit the file */usr/lib/X11/xdm/Xsession.dt* (or *Xsession* if not using the IndigoMagic desktop) and make the following change.

Gives anyone on any host access to this display /usr/bin/X11/xhost +

to

restrict access to this host /usr/bin/X11/xhost -

7) Save the file.

8) Remove any '**xhost +**' from the files */usr/lib/X11/xdm/Xsession**

9) Remove any '**xhost +'** from users private *.xsession* files

10) Remove any */etc/XO.hosts* or */etc/X<n>.hosts* files.

11) Ensure the proper permissions and ownership on the following important X configuration files. Use the **chown** and **chmod** commands to adjust accordingly.

Permissions

Permissions	owner	group	file
-r--r--r--	root	sys	/usr/lib/X11/xdm Xservers
-rwxr-xr-x	root	sys	/usr/lib/X11/xdm/Xlogin
-rwxr-xr-x	root	sys	/usr/lib/X11/xdm Xreset
-rwxr-xr-x	root	sys	/usr/lib/X11/xdm/Xstartup
-rwxr-xr-x	root	sys	/usr/lib/X11/xdm/Xstartup-remote
-r--r--r--	root	sys	/usr/lib/X11/xdm/xdm-config
-rwxr-xr-x	root	sys	/usr/bin/X11/X
lwxr-xr-x	root	sys	/X11/Xsgi
-rwxr-xr-x	root	sys	/usr/bin/X11/xdm
-rwxr-xr-x	root	sys	/usr/bin/X11/xauth
-rwxr-xr-x	root	sys	/usr/bin/X11/xhost

12) Restart the graphics system.

usr/gfx/stopgfx; usr/gfx/startgfx &

- */etc/config/ypbind.options* contain the **-ypsetme** option by default. This will allow someone who can fake your ip address to change your YP binding. To close the hole remove the **-ypsetme** option and add the **-s** option, that will give you a little extra protection.
- Add a **"-a"** to the **rlogind** and **rshd** line in */etc/inetd.conf* to require remote hostnames and addresses to match. If by any chance you want to disallow *.rhosts* files you can add the **"-l"** flag. Please be aware that this will deprive you of real functionality.
- NFS connections to unprivileged ports are accepted by default. Set the **"nfs_portmon"** kernel parameter to 1 to reject NFS connections to unprivileged ports.
- xdm looks for X terminal login requests on port 177 by default. This could be undesirable in some environments. To restrict this access edit */var/X11/xdm/Xaccess* and place a **"!"** in front of each of the two lines that begin with an asterisk to prevent all XDMCP requests.

- */etc/init.d/rmtmpfiles* resets the permissions on */tmp* and */var/tmp* at every bootup. Permissions are set to **1777**, by default; the 1 means sticky, that means no user can remove another's files in the temporary directories. Having this directory to 777 can open your system to some vulnerabilities. Please take a look at :
<ftp://ciac.llnl.gov/pub/ciac/bulletin/f-fy95/f-27.permissions-on-tmp.asc>
- The default root crontab in current IRIX (*/var/spool/cron/crontabs/root*) creates the SYSLOG and cron log with group and world read permission. Change the "033" on lines 25 and 27 to "077" to prevent non-superusers from reading this files.
- Non-root users can give away files. This can be used to defeat accounting and quotas. Set the "**restricted - chown**" kernel parameter to 1 to allow only root to give away files. Be aware this may affect programs like */bin/mail*, that depend on unrestricted *chown* when delivering to an NFS volume without root access.
- Some unnecessary services are enabled by */etc/inetd.conf*. The "echo" and "chargen" services can allow a denial-of-service attack. This is described in CERT Advisory CA-96.01, which can be acquired at:

ftp://ftp.cert.org/pub/cert_advisories/CA-96.01.UDP_service_denial

To disable the "echo" and "chargen" services, comment out the lines which begin with their names in the */etc/inetd.conf* and the execute the "*killall HUP inetd*" command. You may want to examine and consider disabling the unused UDP-based services as well. As shipped from the factory, the IRIX operating system environment permits a fairly wide range of services through *inetd(1M)*. Sites should reduce the available services by editing */etc/inetd.conf* per the manual pages and refreshing *inetd* with the new configuration via "*killall -HUP inetd*". To remove a service, either comment the service out with a '#' character as the first character of the line, or remove the service line entirely from the file. Services left accessible can be configured to improve security by using certain options. Below, some options to consider are listed, but the manual pages should be referred to for completeness.

rlogind use '-1' disable validation using *.rhosts* files

fingerd use '-1' to log all connections
use '-S' to suppress information about login status, home directory, and shell
use '-f msg-file' to make it just display that file

rshd use 'a' to verify that all incoming remote host names and addresses match
use '-1' to disable validation using *.rhosts* files
use '-L' to log all access attempts to syslog

For standard logins, it is prudent to enhance security with several options as described in the manual pages for **login**, **login(1)**.

```
login    set MANDPASS=YES
          set SYSLOG=ALL
          set LOCKOUT=5
```

- Several devices have permissions which might allow a user to monitor another user via audio or video input, to include the following:

```
/dev/audio
/dev/dmrb
/dev/hdsp/*
/dev/vid
/dev/video
```

One of the solutions (Bill Paul of ctr.columbia.edu) is to add the following to */usr/lib/X11/xdm/Xstartup*:

```
chmod 600 /dev/audio /dev/hdsp/* /dev/video /dev/vid /dev/dmrb chown
$USER /dev/audio /dev/hdsp/* /dev/video /dev/vid /dev/dmrb
```

Also add the following to */usr/lib/X11/xdm/Xreset*:

```
chmod 600 /dev/audio /dev/hdsp/* /dev/video /dev/vid /dev/dmrb
chown root /dev/audio /dev/hdsp/* /dev/video /dev/vid /dev/dmrb
```

5.8 X Windows Security

Access to your X server may be controlled through either a host-based or token-based authentication method. Host authentication is the potential acceptance of a connection based on its origin. The token method is to verify each client based on the token they offer. The former is left to the discretion of the Systems Administrator at your site and is useful as long as all hosts registered in the */etc/Xn.hosts* file have users that can be trusted, where "n" represents your X server's number.

This may not be possible at every site, so a better method is to educate each and every user about the security implications (see references below). Better still, when setting up a user, give them a set of X security related template files, such as *.xserverrc* and *xinitrc*. These are located in the users home directory.

Some of the threats to which Xwindows is exposed arise from the fact that any client that can access and change any X communications that take place on the server. A list of the most common threats are:

- Modification of session parameters.
- Creation or destruction of windows where work is being performed.
- Capture X events, reading keystrokes on an xterm window, which include login and password.
- Create X events, for example, sending keystroke sequences to an emacs window, or an xterm window, to execute a command.
- Viewing the contents of your screen remotely.

You are advised to read the section on X windows security referred to in Appendix D.

5.8.1 Problems With xdm.

Security risks that xdm poses:

- Badly configured xdm can give access to the machine via accounts without passwords, e.g., sync on Suns.
- Xdm gives you a sense of security that does not exist:
 - a. by default xdm allows key snooping while you type your password.
 - b. even an AFS-ware xdm sends your password in clear text over the network.
- In some instances, when xdm cannot write into the home directory, the MIT magic cookie is put in */tmp*.

Release 6 of X11 is now available and solves many problems associated with X security which were present in previous releases. If possible, obtain the source for R6 and compile and install it on your system. R6 is available via anonymous ftp from: <ftp://munnnari.oz.au/X.V11/R6>. *xdm* bypasses the normal getty and login functions, which means that quotas for the user, ownership of */dev/console* and possibly other preventive measures put in place by you may be ignored. You should consult your vendor and ask about potential security holes in xdm and what fixes are available.

5.8.2 X Security -Tools

a. Read the man pages for *xauth* and *Xsecurity*. Use this information to set up the security level you require. The *xauth* command is widely used to avoid unauthorized connections to your Xdisplay. The login process goes as follows:

1. The user logs in and xdm creates a file called *.Xauthority* in their home directory.
2. *Xauthority* is a binary file only readable through the *xauth* command. If you issue the following command:

\$xauth list

You should see the following output:

```
your.display.ip:O MIT-MAGIC-COOKIE <key>  
display name      authorization type key
```

If you want to open your display for connection from a particular user, you must inform him/her of your key. The other user must in return issue the following command:

```
$xauth add yourdisplay. ip:O MIT-MAGIC-COOKIE <key>
```

Now only that specific user and you can connect to your display. See Appendix D for a Procedure on How xauth works and how to set it up.

b. Check that the permissions on /tmp are set to 1777 (or drwxrwxrwt); i.e., the sticky bit should be set. The owner must always be root and group ownership should be set to group-id 0, which is "wheel" or "system". If the sticky bit is set, no one other than the owner can delete the file /tmp/.X11-unix/X0, which is a socket for your X server. Once this file is deleted, your X server is gone forever! If the permissions or ownership are not set as above, then type the following commands:

```
#!/bin/chown root.wheel/tmp  
#!/bin/chmod 1777/tmp
```

This will not recursively set the sticky bit on sub-directories below /tmp, such as /tmp/.X11-unix and /tmp/. NeWS-unix; you may have to set these manually or through the system startup files.

Appendix A - Security Tools

There are many good tools available for checking your system. The list below is not a complete list, and you should not rely on these to do all of your security administration work for you; these tools are intended to assist the System Administrator and to be a guide to additional security administration measures.

A. Useful Tools

1. Crack

Crack is a fast password cracking program designed to assist site administrators in ensuring that users use effective passwords.

2. COPS and Tiger

These packages identify common security and configuration problems. They also check for common signs of intrusion. Though there is some overlap between these two packages, they are different enough that it may be useful to run both. COPS is a collection of short shell files and C programs that perform checks of a system to determine whether certain weaknesses are present. Checks for the following:

- Any world writable */etc/cshrc* or */etc/profile* files
- Default vendor accounts with no passwords
- Bad permissions
- Malformed config files
- Easily guessed passwords
- */dev/kmem* and other devices for read/writability
- Duplicate user ids, invalid field in the password file
- Duplicate group ids, invalid fields in the group file
- Changes in the setuid status of programs in the system
- Bad "root" paths, like a "." in roots path
- Format of the */etc/password* file
- NFS file systems exported to the world

Tiger is a set of scripts that scan a UNIX system looking for security problems. It is designed to hopefully be easy to use, easy to understand and easy to enhance. Tiger has one primary goal: report ways root can be compromised. It checks cron entries, mail aliases, NFS exports, inetd entries, PATH variables, rhosts and .netrc files, access permissions, unusual files, binary alterations. Both are available via *anonymous ftp* from:

[ftp://nasirc.hq.nasa.gov/toolkits/UNIX/Cops 104/](ftp://nasirc.hq.nasa.gov/toolkits/UNIX/Cops%20104/)*

<ftp://nasirc.hq.nasa.gov/loothits/TAMU/>*

3. npasswd and passwd+

These programs are proactive password checkers. They run a series of checks on passwords at the time users set them and refuse password that fail the tests. Note that these programs are not designed to work with shadow password systems. The npasswd command has the following capabilities:

- Configurable minimum password length.
- Configurable to force users to use mixed case or digits and punctuation.
- Checking for "simple" passwords such as repeated letter.
- Checking against the host name and other host-specific information.
- Checking against the login name, first and last names, and so on.
- Checking for words in various dictionaries, including the system dictionary.

Both are available via *anonymous ftp* from:

npasswd:

<ftp://ftp.auscert.org.au/pub/mirrors/ftp.cc.utexas.edu/npasswd/npasswd.tar.Z>

passwd+:

<ftp://ftp.auscert.org.au/pub/mirrors/dartmouth.edu/security/passwd+.tar.Z>

4. tcp-wrapper

This software gives logging and access control to most network services. Allows the System Administrator to monitor and filter incoming requests for servers started by inetd. The System Administrator can selectively deny or allow access to your sites from other hosts on the Internet. It is available via anonymous ftp from:

[ftp://nasirc.hq.nasa.gov/toolkits/UNIXITCP Wrappers 7.2/](ftp://nasirc.hq.nasa.gov/toolkits/UNIXITCP%20Wrappers%207.2/)*

5. Tripwire

This package maintains a checksum database of important system files. It can serve as an early intrusion detection system. It is available via anonymous ftp from

<ftp://nasirc.hq.nasa.gov/toolkits/UNIX/Tripwe/>*

6. cpm

cpm checks to see if your network interfaces are running in promiscuous mode. If you do not normally run in this state then it may be an indication that an intruder is running a network sniffer on your system. This program was designed to run on SunOS 4. Lx and may also work on many BSD systems. It is available via anonymous ftp from:

<ftp://nasirc.hq.nasa.gov/tools/UNIX/CPM/>*

7. Vendor supplied C2 Security Packages

Consult manuals supplied by your vendor as to installing C2 security. The SunOS manual is "SunOS System & Network Administration Guide".

8. Vendor Supplied Security Auditing Packages

Sun provides an additional security package called SUNshield. Please direct esquires about similar products to your vendor.

9. smrsh

The smrsh(8) program is intended as a replacement for */bin/sh* in the program mailer definition of sendmail(8). *smrsh* is a restricted shell utility that provides the ability to specify, through a configuration, an explicit list of executable programs. When used in conjunction with sendmail, smrsh effectively limits sendmail's scope of program execution to only those programs specified in *smrsh's* configuration.

10. MD5

MD5 is a message digest algorithm.

11. SATAN (Security Analysis Tool for Auditing Networks)

This security tool remotely scans and reports various bugs and weaknesses in network services and windowing systems, as well as detailing as much useful information as possible about the target machine. It will process the data with a data filtering/interpreting program that generates the final security analysis.

B. More Tools

1. Binaudit

This package audits local and remote systems and compares the audit listing with previously generated listings that reveal what had changed. Characteristics of some system files are expected to change occasionally and should not be reported, but ownership and access protection on these files should. Binaudit allows the system administrator to customize the audit to his or her site. After the auditing information has been collected the results of the audit are to be sent. This tool is useful to the system administrator by pointing out specific changes to files that are critical to the systems security and integrity. Available from:

<ftp://nasirc.hq.nasa.gov/toolks/UNIX/Binaudit/>*

2. Courtney

Monitors the network and identifies the source machines of SATAN probes/attacks. Courtney receives input from tcpdump counting the number of new services a machine originates within a certain time window. If one machine identifies the machine as a potential SATAN host. Available from:

<ftp://nasirc.hq.nasa.gov/toolkits/UNIX/Courtney/>*

<ftp://ciac.llnl.gov/pub/ciae/sectools/unix/courtney/>*

3. MD5check

A valuable check summing utility, available for systems running SunOS 4X MD5check compares the MD5 checksums of several critical SunOS 4.X system files to a database of known good checksums and then generates a report indicating any discrepancies. This program is useful in finding Trojan binary files or files that are not in the checksum table. A correct match indicates that the file is free from tampering. Available from:

<ftp://nasirc.hq.nasa.gov/toolkits/UNIX/Checksums/Md5check/>*

4. Digger

"Domain Internet Gropper" is a utility for querying domain nameservers. The tool is very useful for tracing an IP address to its proper name and place of origin. Available from:

<ftp://nasirc.hq.nasa.gov/toolkits/UNIX/Digger>

5. Ethernian

This is an X11 based package that can graphically depict Ethernet connections in real time. It is useful tool for providing statistical information in a graphical manner on how the network is being used. It will also provide protocol summaries, usage statistics and snapshots of network summaries. Available from:

ftp://nasirc.hq.nasa.gov/toolkits/UNIX/Etherman/*

6. Fremont

The "fremont" network explorer uses a variety of protocols to help system administrators discover key network characteristics such as hosts, gateways, and topology. It runs on SunOS 4.X. This utility provides information on a wide variety of protocols and information in a timely manner recording information in a database. This database can be used to highlight inconsistent information through out the network. Available from:

ftp://nasirc.hq.nasa.gov/toolkits/UNIX/Fremont/*

7. Gated

The "gated" routing daemon is more secure than the standard "vanilla" UNIX distribution routing daemon(s). The router Information Protocol (RIP) will refuse to run if it determines that the UDP checksums are disabled in the kernel. Available from:

ftp://nasirc.hq.nasa.gov/toolkits/UNIX/Gated/*

8. Interman

Interman is an X11 based package that can graphically depict network connections on your local network segment in real time. It keeps the system administrator informed and protocol activity on the network. Available from:

ftp://nasirc.hq.nasa.gov/toolkits/UNIX/Interman/*

9. LARCscan

The LaRCSCAN non-intrusive network security scanner package, developed at the NASA Langley Research Center, is useful for pointing out vulnerabilities in the networked UNIX systems. This utility should be run in the background due to the amount of time needed to run. Available from:

<ftp://nasirc.hq.nasa.gov/toolhits/UNIX/LARCscan/>*

10. Lastlog

Lastlog is a NASIRC developed utility that can interactively dump your system log files in "plain English" format and flag inconsistencies. It is intended to aid system maintenance, to detect unusual account activity, and to detect tampering with the lastlog file. Available from:

<ftp://nasirc.hq.nasa.gov/toolhits/UNIX/Lastlog/>*

11. Naiad

The NASIRC Automated Inode Anomaly Detector (Naiad) utility scans file systems for unique and/or unusual file names, inode changes, etc. that might indicate sniffing or other undesirable activity. Its purpose is to find evidence of attempted or actual system tampering. Naiad is intended to be used in conjunction with checksumming programs. Available from:

<ftp://nasirc.hq.nasa.gov/toolhit/UNIX/Naiad/>*

12. NSFWatch

The "*nsfwatch*" utility helps monitor NFS requests to any given machine (or an entire network); it can monitor NFS client traffic as well as NFS reply traffic from a server to measure RPC response time. This tool has been compiled and tested on the following architectures and operating systems:

Architecture	Operatin System
Sun-3 (68000)	SunOS 4.1.1
Sun-4 (SPARC)	SunOS 4.1.1, 4.1.2, 4.1.3
Sun-4 (SPARC)	SunOS 5.1, 5.2, 5.3
DEC VAX	Ultrix 4.0, 4.1, 4.2
DEC RISC	Ultrix 4.0, 4.1, 4.2
DEC Alpha AXP	DEC OSF/I V1.3 and later
SGI Personal IRIS	IRIX 4.0.1
SGI 4D/440	IRIX 4.0.5

Available from:

<ftp://nasirc.hq.nasa.gov/toolkits/UNIX/NFSwatch/>*

13. Packetman

This tool allows the capture and analysis of an Ethernet packet trace.

14. Portmapper-3

Provides a simple mechanism to discourage access to NIS), NFS, and other services registered with the portmapper. It can prevent undesirable client-server interactions and remote connections. Available from:

<ftp://nasirc.hq.nasa.gov/toolkits/UNIX/Portmap>
<ftp://info.cert.org/pub/tools/nsf tools/>*

15. RDIST

A version of the "rdist" daemon that is more secure than the standard version (it does not need to run as root, thereby giving crackers one less potential hole to exploit). This version supports automatic system administration determination and configuration for pre-ported platforms. Available from:

<ftp://nasirc.hq.nasa.gov/toolkits/UNIX/RDIST/>*

16. Sendmail

The latest version of the sendmail program for a variety of UNIX systems. This version contains patches to the sendmail program which make it more secure. All previous versions of sendmail are obsolete and have security vulnerabilities. Available from-

<ftp://nasirc.hq.nasa.gov/toolkits/UNIX/Sendmail/>*

17. Slic

Scan for log file inconsistencies, another NASIRC developed tool, scans several system logfiles for inconsistencies that could indicate unauthorized intrusions to your system. Slic was inspired by the discovery of tools such as "invisible" which help to cover the trails of UNIX intruders by altering logfiles. Available from:

<ftp://nasirc.hq.nasa.gov/toolkits/UNIX/Slic/>*

18. Swatch

The "Simple WATCHer and filter" (SWATCH) package assists the system administrator from being overwhelmed by large amounts of system log information on several systems. Swatch monitors log files and acts to filter out unwanted data and notify the system administrator based on patterns in the logs. Available from:

<ftp://nasirc.hq.nasa.gov/toolhits/UNIXISwatch/>*

19. TCPdump

The "tcpdump" network monitoring and data acquisition tool is a protocol packet and dumper program to analyze network traffic. This tool is useful in capturing packets on a network if you suspect and suspicious activity. Tcpcdump has been built and tested under SGI Irix 4.x and 5.2, SunOS 4.x, Solaris 2.3, BSD/386 v1.1, DEC/OSF v1.3, v2.0, and Ultrix 4.x. Available from:

<ftp://nasirc.hq.nasa.gov/toolhits/UNIX/TCPdump/>*

20. Traceroute

A system administrators utility to trace the route IP packets from the current system take in getting to some destination system. Traceroute is useful in determining the path of an IP address through networks and routers and showing any delay in the network pattern. This program requires the following:

- That root runs it (it uses raw ip sockets).
- Requires a Kernel Mod to the raw ip output code to run.

Available from:

<ftp://nasirc.hq.nasa.gov/toolkits/UNIX/Traceroute/>*

21. Thumb

A NASIRC developed utility that can list the users on a system more accurately than "finger". The letclutmp file is actually world-writable on some versions of UNIX such as SunOS 4.1.X. This makes it trivial for any users to erase their entries or provide misleading information. Intruders have been known to take advantage of this to hide their presence. Thumb bypasses the /etc/utmp file by process information from the 'ps' program. Thus thumb is invulnerable to a badly-updated or altered /etc/utmp. Available from:

<ftp://nasirc.hq.nasa.gov/toolhits/UNIX/Thumb/>*

22. **Watcher**

Uses system utilities to provide long term tracking or potential problems in individual UNIX systems. Watcher is run at suitable intervals and e-mail status reports. The watcher tool introduces some intelligence into the machine's self-reporting, letting the machine filter out messages indicating normal operation and forwarding only those messages which point out trouble areas. The result of these efforts is watcher, a very general and extensible system self-monitor. Watcher keeps closer tabs on the system; since it delivers only a summary of potential problems. No problems slip by unnoticed in the more concise output, leading to an improvement in overall system availability as well as the more effective utilization of the system manager's time. Available from:

ftp://nasirc.hq.nasa.gov/toolkits/UNIX/Watcher/*

Appendix B - References

1. Practical UNIX Security, Simson Garfinkel and Gene Spafford; 1991 O'Reilly & Associates, Inc.
2. UNIX Systems Security, Patrick Wood and Stephen Kochan; 1986 Hayden Books.
3. UNIX System Security: A Guide for Users and System Administrators, David A. Curry; May 1992 Addison-Wesley Professional Computing Series.
4. X Windows System Administrators Guide, Chapter 4; 1992 O'Reilly & Associates, Inc.
5. Information Security Handbook, William Caelli, Dennis Longley and Michael Shain; 1991 MacMillan Publishers Ltd.
6. Firewalls and Internet Security, William R. Cheswick & Steven M. Bellovin; 1994 AT&T Bell Laboratories, Addison-Wesley Publishing Company.
7. CERT advisories can be found via anonymous FTP from <ftp:Hinfo.cert.org/pub/cert-advisories/>*
8. UNIX System Administration Handbook, Nemeth, Evi, Garth Snyder and Scott Seebas; 1989 Prentice-Hall, Englewood Cliffs(NJ).
9. Essential System Administration, Aeleen Frisch; O'Reilly & Associates, Inc.

Appendix C - Shell Scripts and Useful Commands

1. Script for printing the umask value for each user:

```
#!/bin/sh PATH=/bin:/usr/bin:/usr/etc:/usr/ucb
```

```
HOMEDIRS='cat /etc/passwd | awk -F`:"length($6) > 0 {print $6}}' | sort -u  
FILES=".cshrc login profile"
```

```
for dir in $HOMEDIRS do  
for file in $FILES do grep-s umask /dev/null $dir/$file  
done  
done
```

2. Command to look for setuid and setgid files

a. The unix find program can be very useful in finding setuid and setgid files:

```
# find /-user root -perm -4000 -print          setuid  
# find /-user kmem -perm -2000 -print        setgid  
# find /-perm -4000 -o -perm -2000 -exec ls -ld {} \;
```

The above commands search the entire directory tree, some find commands support an `-xdev` option to avoid this search:

```
# find/ -user root -Perm -4000 -print -xdev
```

b. The `ncheck` command can also be used:

```
# ncheck -s /dev/rsdOg
```

Appendix D - xauth: How it works and Setup Instructions

xauth works by creating a file called Xauthority in your home directory. This file contains display names followed by a hexadecimal number (the magic cookie). The X server reads the magic cookie from this file and then requires any X program you run to provide the same magic cookie.

Your X client programs also read the Xauthority file to obtain the magic cookie. The client sends the magic cookie to the server, then the server verifies that it matches the one in memory. If they match, the client is allowed to open a window on the X server.

If you are running X programs on the same machine your X server is on, then the security is trivial, because both programs have access to the Xauthority file. The Xauthority file permissions are set to 600, theoretically preventing anyone else on your system from reading your magic cookies.

If you need to run X programs on a remote host, you must first propagate the magic cookie for your local display to the remote host. **xauth** and **X** provide ways of doing this.

Setting it up

Step 1

The first step is to create an Xauthority file with a magic cookie for your host. This is easiest done in your `.login` or `.kshrc` file. This code can also be included in an `.xseruerrc` file.

The `xauth` program creates and manages the Xauthority file. Make sure `/usr/bin/X11` (or wherever your X binaries are) is in your path. It is best to set the X path in your startup files, either `.profile`, `.kshrc`, or `.cshrc` since these files are read for `rsh` commands. You'll be using `rsh` to execute X programs on remote hosts. Here is some sample code using `perl` and `sh` or `ksh` for your Korn Shell or Born Shell `.profile`:

One of the best ways we have found to generate a random non-reproducible number with output in hex is to use common system commands and the MD5 cryptographic sum generator. MD5 can be retrieved from ftp.cert.org.

A `csh` example using system statistics and MD5.

```
set randomkey='(ps -ael; nfsstat -m; nfsstat -r; netstat -s; netstat -m; date)' | md5
xauth add 'hostname'/unix:O . $randomkey
xauth add 'hostname':O . $randomkey
```

A **sh** or **ksh** equivalent would be:

```
randomkey=$( (ps -ael; nfsstat -m; nfsstat -r; netstat -s; netstat -m; date) | md5);  
xauth add $(hostname)/unix:O. $randomkey  
xauth add $(hostname):O - $randomkey
```

Note: these command line options should work with BSD and System V variants. A **csh** example using **perl**.

```
set randomkey='perl -e 'for (L.4) {\  
srand(time+$$+$seed); \  
printf( "'%4.5x'", ($seed = int(rand(65536)))));}\  
print "\n";'
```

```
xauth add 'hostname'/unix:O . $randomkey  
xauth add 'hostname':O . $randomkey
```

An important note: the magic cookie must be a string of hexadecimal digits that is an even length. MD5 does this automatically. The benefits here are:

1. A new magic cookie is generated each time you log in and once installed it is virtually transparent
2. "**xauth add**" stores the magic cookie in your Xauthority file in your home directory, and ensures the permissions are **-rw ------** restricting access to the file.

Step 2 - enabling **xauth** authentication

Now that you have a Xauthority file with your magic cookie in it you need to tell the X server to use this to authenticate X programs that want to use your display. You can do this by passing the **-auth** argument when starting the X server.

If you are using **xinit** in your Jlogin you would change the line to something like:

```
xinit $HOME/.xinitrc -- /usr/bin/X11/X -auth $HOME/.Xauthority
```

It is important to use the **--** argument to **xinit**, this tells **xinit** to pass the following arguments on to the X server. Something like:

```
xinit $HOME/.xinitrc - -auth $HOME/.Xauthority  
will probably work, while this:
```

```
xinit $HOME/.xinitrc -auth $HOME/.Xauthority
```

will not.

If you use a *xserverrc* file to start your X server, add the **-auth** option:

```
exec /usr/bin/X11/X -auth $HOME/.Xauthority
```

Once the X server is started with the **-auth** parameter, it will use the magic cookies to verify programs wanting to use your X display. Unless there are still hosts uthorized with xhost.

Step 3 - clear xhost's authorization list

xhost supersedes **xauth** so if you still have hosts enabled under xhost then the magic cookie scheme will do you no good.

Check your */etc/XO.hosts* to see which hosts will be automatically authorized, or put a

xhost -

in your. login or equivalent file. This will effectively disable xhost authorization, passing all authorization on to **xauth** with magic cookies.

The X Windows System Administrator's Guide warns against totally removing or disabling xhost because this could be equivalent to having **xhost +**, allowing access to all hosts. Using "**xhost -**leaves **xhost** enabled, but empties the list of authorized hosts so all authorization happens through **xauth** and magic cookies.

At this point, you should be secure to operate X on a single workstation. You should be able to run any clients from your host and they will read your *.Xauthority* file. If your home directory is shared among many hosts (ex. through NFS mounting), you can run clients from any of those hosts, because they will all have access to the *Xauthority* file in your shared home directory.

Step 4 - running clients on remote hosts (**xauth/rsh**)

In order to run clients on remote hosts, you must have a way of getting the magic cookie to those clients so they can authenticate themselves to your local X server. **xauth** provides a way to create a *Xauthority* file on your remote host so remote clients will be able to read the magic cookie.

You must first make certain that you have your local host in the. *rhosts* or */etc/hosts*. *equiv* on the remote machine, otherwise rsh will not work. Also make sure the path to **xauth** (usually */usr/bin/X11*) is set in the *cshrc* on the remote host.

Now you are ready to propagate the magic cookie to the remote host. For example, if you're on host ren and you want to run a client on stimp you would first type:

```
ren% xauth extract - ren:O | rsh stimp xauth merge -
```

This pipes the magic cookie to xauth on stimp which builds a new *.Xauthority* file with the magic cookie for display ren:O. Now you could execute:

```
ren% rsh stimp /usr[bin/X11/xterm -display ren:O
```

and xterm on stimp would read the new *Xauthority* and find a magic cookie allowing it to run on ren's display.

Step 5 - automation with xrsh

The above steps have been combined in a system called xrsh which is available as */contrib/xrsh-5.1shar* at ftp.x.org. xrsh automatically does the **xauth** stuff for you, as well as always using csh since ksh doesn't read any init files for a rsh, leaving you without a path set. xrsh provides a **xrlogin** command as well as some other security precautions. We highly recommend installing it.

Using xrsh, and assuming the path to the X binaries is set in the *CShrc* on stimp, the above would become:

```
ren% xrsh -auth xauth -pass ENV stimp xterm
```

Be sure to include the **-auth xauth** because **xrsh** defaults to use xhost authentication. You could alias xrsh to include **-auth xauth -pass ENV**:

```
ren% alias xrsh'xrsh -auth xauth -pass ENV \!*
```

and then use:

```
ren% xrsh stimp xterm
```

That should allow you to accomplish anything that was previously done with xhost. Furthermore, you can still use xhost if you feel the need, although this instantly compromises any xauth security you have set up. If you feel the need to use **xhost** we recommend issuing a

```
% xhost -
```

immediately after finishing whatever required xhost, or simply logging out and logging back in, which will generate a new magic cookie and restart X with **xauth** authorization reset.

Appendix E - Abbreviated Checklist

It is intended that this short version of the checklist be used in conjunction with the full checklist as *Certification Documentation*. To be used as Certification Documentation, the UNIX Administrator shall complete *column b* of the checklist and sign and date the second page of the checklist. Also, the System Owner shall sign and date the second page of the checklist.

The numbers of the checklist items below correspond to the paragraph numbers in the *Practices* section of this guide.

UNIX Guideline Abbreviated Checklist

UNIX Checklist Items	Implement Yes / No	Remarks
A. Patches		
Installed latest patches?		
B. Network Security		
1. Filtering		
2. <code>nc</code> commands		
3. <code>/etc/hosts.equiv</code>		
4. <code>\$HOME.rhosts</code>		
5. NFS		
6. <code>/etc/hosts.lpd</code>		
7. <code>/etc/ttytab</code>		
8. <code>/etc/inetd.conf</code>		
9. Trivial ftp (tftp)		
10. <code>/etc/services</code>		
11. <code>tcp_wrapper</code> (also known as <code>log tcp</code>)		
12. <code>/etc/aliases</code>		
13. <code>/etc/sendmail.cf</code>		
14. <code>majordomo</code>		
15. <code>fingerd</code>		
16. UUCP		
17. WWW		
a. Is your Web Server on a dedicated machine?		
b. Are you running your Server as root?		
c. Automatic directory listings disabled.		
d. Are all login available necessary?		
e. Are you NFS mounting or exporting		

UNIX Checklist Items	Implement Yes / No	Remarks
any directories?		
f. Are all services running necessary?		
g. Are all logging files being used? access - log, agent_log, error-log, refer log		
4.3 ftpd and anonymous ftp		
1. Versions		
2. SITE EXEC		
3. Configuration of your ftp server		
4. Permissions		
5. Writable directories		
6. Disk mounting		
D. Password and Account Security		
1. Policy		
2. Proactive Checking		
3. Root Password		
4. NIS and /etc/passwd entries		
5. Password shadowing and C2 security		
6. Administration		
7. Special accounts		
8. Root account		
E. File System Security		
1. General		
2. /etc/rc.local		
3. /usr/lib/expreserve		
4. External file systems/devices		
5. File Permissions		
6. Files run by root		
7. Bin ownership		
8. Tige		
9. COPS		
F. SUNOS Specific Security		
1. IP forwarding		
2. Framebuffers /dev/fb		
3. /usr/kvm/sys/*		
4. /dev/nit (Network Interface Tap)		
G. IRIX Specific Security		

UNIX Checklist Items	Implement Yes / No	Remarks
1. /usr/lib/vadmin/serial_ports		
2. IRIX Patches		
H. X-Windows Security		
1. Problems with xdm		
2. X security - General		

Machine Identification and Location

UNIX Administrator

Date

Security Analyst

Date

Major Security Exeptions and justification:

NASA HQ ITS Manager

Date