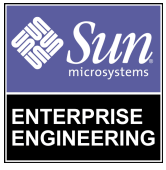
Solaris™ Operating Environment Security

By Alex Noordergraaf and Keith Watson - Global Enterprise Security Service

Sun BluePrints™ OnLine - January 2000



http://www.sun.com/blueprints

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 806-4393-10
Revision 01, January 2000

Please
Recycle

Solaris™ Operating Environment Security

The Solaris Operating Environment is a flexible, general purpose operating system. Due to its general nature, changes must be made to secure the system against unauthorized access and modification. This article discusses the Solaris Operating Environment subsystems and the security issues surrounding those subsystems. Recommendations are made on the manner in which those subsystems should be secured.

As with any security decisions, a balance must exist between system manageability and security. Some changes in this article will not be applicable to all environments. The removal of some of the Solaris Operating Environment services mentioned in this article may negatively impact the ability to effectively maintain a system. You must know your system and security requirements before starting.

This article splits the discussion of the Solaris Operating Environment system security into two parts. The first section deals with file system and local security. The second half of this article deals with the security of network services.

The information in this article applies to the Solaris 2.5.1, 2.6, and 7 Operating Environment. Some system changes described in this article have been integrated into Solaris 8 Operating Environment. Where appropriate, these new changes are mentioned. Older versions of the Solaris Operating Environment may be configured in similar ways. Some investigation is necessary before making the changes suggested in this article to these older versions.

File Systems and Local Security

It is important that the file systems and local security of the Solaris Operating Environment system not be neglected. Often, administrators are greatly concerned about attackers breaking into systems remotely. There should be equal concern for local, authorized users gaining extra privileges on a system by exploiting a problem with internal system security.

Initial Installation

Building a secure Solaris Operating Environment system involves installing a new system with the latest version of the Solaris Operating Environment and applying the latest patches.

Solaris Operating Environment Installation

Sun works toward improving the Solaris Operating Environment with every release. Each new release includes security improvements and additional features to enhance system security. Always use the latest version of the Solaris Operating Environment that your applications will support.

To prevent an attacker from modifying a system or creating backdoors before you have the opportunity to secure it, perform an initial Solaris install. Do not perform an upgrade to an existing Solaris system. Also, install the system from an original Sun Solaris Operating Environment CD, and do not attach the system to a “public” network until the modifications have been made.

Partitions

When creating operating system file partitions, be sure to allocate adequate disk space for system directories, log files, and applications. Certain server applications or services may require extra disk space or separate partitions to operate effectively without impacting other services. Typically, there should be separate partitions for the root file system (`/`), `/usr`, `/var`, and `/opt`.

The Solaris Operating Environment `/var` file system contains system log files, patch data, print, mail, and files for other services. The disk space required for these files will vary over time. Most systems (and all servers) should maintain `/var` as a separate partition from the root file system. Mail servers should maintain a large,

separate /var/mail partition to contain user mail files. These extra partitions will help prevent a full /var or /var/mail file system from affecting the operation of the system. Provide extra space in /var if you intend to store large log files.

Most applications install themselves in /opt or /usr/local. Check the application installation directory location before allocating space.

Minimization

It is important to reduce the Solaris Operating Environment installation down to the minimum number of packages necessary to support the application to be hosted. This reduction in services, libraries, and applications helps increase security by reducing the number of subsystems that must be disabled, patched, and maintained.

The December 1999 Blueprints Online issue includes an article entitled “Solaris Operating Environment Minimization for Security” which describes a methodology for the minimization and automation of Solaris Operating Environment installations.

Patches

Sun provides patches to the Solaris Operating Environment and unbundled software products when problems are corrected. Anyone can download recommended, security, and Y2K patches for the Solaris Operating Environment. All other patches require a SunSpectrum service contract. All systems should have the latest recommended, security, and Y2K patches installed. Subscribe to the Sun security bulletin mail list to receive notification of important security related patches. Recently, Sun started providing Maintenance Updates (MU) for the Solaris Operating Environment. A MU is a tested combination of patches for a specific release of the Solaris Operating Environment that installs in one quick and easy step. These updates are only available to service contract customers.

SunSpectrum service contract customers have access to all patches, maintenance updates, and the patchdiag tool. patchdiag takes a list of current patches available from Sun and examines the local system to determine patches that have not yet been applied. It also checks for new versions of patches that have already been applied. The patchdiag tool should be run on the system at least once a week to determine if important patches need to be applied such as security patches.

Immediately after a Solaris Operating Environment system is installed, all recommended, security, and Y2K patches should be applied. These patches are available from the sunsolve.sun.com Web and FTP sites.

Care must be taken when applying patches to a system. Some patches modify the system initialization scripts and may change the configuration of a system. Scripts that were deleted from the init run level directories to disable services, should be

replaced, enabling the service once more. Be sure to examine all system `init` scripts and test all patches on non-production systems to discover any such configuration changes.

Console Security

There are several security mechanisms that Sun hardware systems provide. The OpenBoot PROM system on SPARC architecture based systems has two security modes. Failed login attempts to the OpenBoot PROM system can be monitored. It is also possible to prevent users from aborting Solaris Operating Environment and dropping to the OpenBoot PROM level using the keyboard abort sequence.

OpenBoot PROM Security Modes

Sun SPARC architecture based hardware provides some additional console security features. These features prevent EEPROM changes, hardware command execution, and even system start-up without the appropriate password. This password protection only works while the system is at the OpenBoot PROM level (Solaris Operating Environment is stopped). Similar features might be available on Intel x86-based hardware, but they have no meaning to the Solaris Operating Environment (Intel Platform Edition).

The OpenBoot PROM password is not related to the Solaris Operating Environment root password. Once set, the password is not displayed, but can be retrieved in clear text form. You should not set the OpenBoot PROM password to the same password as the root password. When changing the OpenBoot PROM password, the system will not ask for the old password prior to changing it to the new one. In some environments it may make more sense to set the OpenBoot PROM password to something known to the hardware technicians.

There are two security modes available. The `command` security mode prevents EEPROM changes and hardware command execution while at the OpenBoot PROM level. The `full` security mode provides the features of the `command` mode and, in addition, the system will not boot without the correct OpenBoot PROM password. Full security mode requires operator interaction to boot the system. It will not boot without a password. Do not use this feature on servers or other systems that must boot quickly without outside intervention.

To set the security mode, use the `eeprom` command in the Solaris Operating Environment. Here is an example of setting the mode to `full`:

```
# eeprom security-mode=full
Changing PROM password:
New password: password
Retype new password: password
```

To set a new EEPROM password, use the following command:

```
# eeprom security-password=
Changing PROM password:
New password: password
Retype new password: password
```

Be sure to include the trailing equal sign (“=”).

These OpenBoot PROM changes can also be made while at the OpenBoot PROM level. Here is an example of setting the OpenBoot PROM security mode and password while at OpenBoot PROM level:

```
ok setenv security-mode command
security-mode =       command
ok setenv security-password password
security-password =
```

The system EEPROM security mode can be disabled by setting the security mode to none.

Monitoring EEPROM Password Guessing

If someone guesses or mistypes the OpenBoot PROM password, a time-out period of ten seconds occurs and the attempt is counted. To see how many bad log attempts have been made use the following command:

```
# eeprom security-#badlogins
security-#badlogins=3
```

You may want to add this command to an initialization script to track password attempts. To reset the counter, use the following:

```
# eeprom security-#badlogins=0
security-#badlogins=0
```

Losing the OpenBoot PROM password requires that you replace the EEPROM. An attacker with superuser access may set the security mode to `full`, set the password to random characters, and reboot the system. The system will no longer boot without the new password. If this happens, it's time to call Sun for a new EEPROM.

Disabling Keyboard Abort

SPARC architecture based systems can drop to the OpenBoot PROM level while the Solaris Operating Environment is running using the keyboard abort sequence. This can be disabled in Solaris 2.6 and 7 Operating Environment. This feature may be useful in uncontrolled lab environments to prevent users from bringing systems down. If the OpenBoot PROM security mode is enabled, the EEPROM settings cannot be altered without a password.

To disable the keyboard abort sequence change the following line from the `/etc/default/kbd` file:

```
#KEYBOARD_ABORT=enable
```

to:

```
KEYBOARD_ABORT=disable
```

Should the system hang or otherwise become unusable, it will have to be powered off to be reset. It will no longer be possible to create a crash dump from the OpenBoot PROM level on a running system for analysis.

File system

The Solaris Operating Environment file system can be adjusted to provide added protection. The default file permissions on some files are not adequate. There are also several mount options that increase security when used effectively. The Solaris Volume Management system needs some adjustment to prevent attackers from gaining superuser privileges.

Adjusting File Permissions

The Solaris Operating Environment ships with some file system permissions that should be adjusted for security reasons. Many files and directories have the group write bit set. In most instances, this permission is not necessary and should be switched off. Casper Dik has created a tool to adjust these permissions. The tool is called `fix-modes`. Please note that this tool is not supported by Sun. The `fix-modes` program must be compiled on a Solaris Operating Environment system with a C compiler. Once compiled, install the `fix-modes` files and execute it to correct file system permissions. This tool has been used in production environments for several years with no reported problems. Be careful when installing patches and new packages. These may set permissions back to the original state. `fix-modes` should be executed after all packages are installed and all patches are applied. Sun is working to integrate the permission changes made by the `fix-modes` tool into new releases, starting with the Solaris 8 Operating Environment.

set-user-ID and set-group-ID files

The `set-user-ID` and `set-group-ID` bits (sometimes referred to as SUID and SGID bits) on an executable file indicate to the system that the executable should operate with the privileges of the file's owner or group. In other words, the effective user ID of the running program becomes that of the executable's owner, in the `set-user-ID` instance. A `set-group-ID` file sets the running program's effective group ID to the executable's group. Often, this means that an executable started by a normal user will operate with superuser privileges. This is useful in allowing users to run some commands that gather system information or write to files not owned by the user. If the command with the `set-user-ID` and/or `set-group-ID` bit set is written correctly with security in mind, this can be a useful method in solving some tricky operational problems.

`set-user-ID` and `set-group-ID` commands that have flaws are often used to exploit the system. The attacker uses the elevated privileges provided by the `set-user-ID` or `set-group-ID` mechanism to execute code on the program stack (a "buffer overflow" attack) or to overwrite system files. When these security problems are reported, Sun fixes them and provides a patch. This is another reason to keep a system up to date with the latest set of patches.

Attackers may also use the `set-user-ID` or `set-group-ID` feature to create backdoors into systems. One way this is done is by copying a system shell to a "hidden" location and adding the `set-user-ID` bit. This allows the attacker to execute the shell to gain elevated privileges (most often superuser).

To find all the `set-user-ID` and `set-group-ID` files on a server use the following find command:

```
# find / -type f \( -perm -u+s -o -perm -g+s \) -ls
```

Store the output to a file on another system. Compare it against the current file system from time to time and after applying patches to find any unwanted additions.

Mount Options

The Solaris Operating Environment file system partitions can be mounted with various options that enhance security. As shown in the previous section `set-user-ID` files can be used by attackers to create ways to gain higher privileges. These backdoors may be hidden anywhere on the file system. While a file may have a `set-user-ID` bit, it will not be effective on file systems mounted with the `nosuid` option. The system ignores the `set-user-ID` bit for all files on a `nosuid` mounted file system, and programs will execute with normal privilege. It is also possible to mount a file system as in read-only mode to prevent file modification. This will also prevent an attacker from storing backdoor files or overwriting and replacing files on the file system. Whenever possible, file systems should be mounted in read-only mode, and should be mounted to ignore the `set-user-ID` bit on files.

It should be noted that these options are not complete solutions. A read-only file system can be remounted in read-write mode. The `nosuid` option can be removed. Not all file systems can be mounted in read-only mode or with `nosuid`. If a file system is remounted in read-write mode, it must be rebooted to switch back to read-only mode. A reboot is also required to change a `nosuid` file system to `suid`. Watch for unscheduled system reboots.

The system partitions support some of these mount options. The `/usr` partition can be mounted read-only. It should not be mounted `nosuid` since there are some commands in this partition that have the `set-user-ID` bit set. The `/var` partition cannot be set to read-only but can be set to `nosuid`. All other partitions should be mounted read-only and with `nosuid` whenever possible.

Contrary to suggestions in other Solaris Operating Environment security documents, it is not possible to mount the root file system (`/`) with the `nosuid` option on modern releases of the Solaris Operating Environment. This is due to the fact that the root file system is mounted read-only when the system boots and is later remounted read-write. When the remount occurs, the `nosuid` option is ignored.

Here is a partial /etc/vfstab file containing the appropriate file system options:

```
/dev/dsk/c0t3d0s0 /dev/rdsk/c0t3d0s0 /    ufs 1 no -
/dev/dsk/c0t3d0s4 /dev/rdsk/c0t3d0s4 /usr ufs 1 no ro
/dev/dsk/c0t3d0s5 /dev/rdsk/c0t3d0s5 /var ufs 1 no nosuid
/dev/dsk/c0t3d0s6 /dev/rdsk/c0t3d0s6 /opt ufs 2 yes nosuid,ro
```

Volume Management

The Solaris Volume Management system provides users an easy way to mount removable media without requiring superuser access. CDROMs and floppy disks are mounted and unmounted automatically by the volume management system. The daemon that manages this system is called vold.

vold uses the rmmount command to mount the removable media device. It uses a configuration file (/etc/rmmount.conf) to determine the actions necessary based on the device to be mounted. vold calls rmmount which determines what type of file system, if any, is on the media. If a file system is present and it is supported, rmmount mounts the file system.

If the system does not require the automatic mounting of CDROMs and floppy disks, Volume Management should be disabled. For example, a server does not need it, but a workstation may. Disabling this service can be accomplished by removing the Volume Management packages (SUNWvolr, SUNWvolu, and SUNWvolg).

If Volume Management is necessary, the mount options for some file systems should be modified for security. As discussed above, file systems with the suid option can be problematic. Unfortunately, the Volume Management system allows suid file systems for all removable media that are capable of supporting it. Anyone can insert a UFS formatted floppy containing a set-user-ID executable and gain control of the system. To prevent this situation, add the following lines to the end of the /etc/rmmount.conf file:

```
mount hsfs -o nosuid
mount ufs -o nosuid
```

With these options, the set-user-ID bit on executables is ignored on file systems that are mounted by the Volume Management system.

Accounts

Managing user and system accounts is an important aspect of the Solaris Operating Environment security. Some system accounts may need to be modified or deleted. The time-based command execution system tools, `cron` and `at`, may also need to be configured to restrict user access.

Managing System Accounts

A default Solaris Operating Environment installation contains several accounts that either need to be deleted or modified to strengthen security. Some accounts are not necessary for normal system operation. These accounts include `smtp`, `nuucp`, and `listen`. Some of these accounts exist to support software subsystems that will not be used or for backwards compatibility. Use the `passmgmt` command to delete accounts in `/etc/passwd` and `/etc/shadow`. Here is an example:

```
# passmgmt -d smtp
```

This command removes the `/etc/passwd` and `/etc/shadow` entries for `smtp`.

The remaining system accounts (except the root account) should also be modified for added security. System accounts listed in `/etc/passwd` have no shell listed. Those accounts also have a `NP` string (meaning “no password”) listed in the `/etc/shadow` file. By default, this is sufficient. However, some additional steps can be taken to add more security. Use the `-l` option of the `passwd` command to lock accounts. To lock the `uucp` account use the following command:

```
# passwd -l uucp
```

Also, use the `-e` option to the `passwd` command or edit the `/etc/passwd` file manually to change the default shell for those accounts to `/usr/bin/true`. For example:

```
# passwd -e uucp
Old shell: /sbin/sh
New shell: /usr/bin/true
```

Some administrators want to monitor these system accounts for abuse. The Titan security package includes a shell replacement called `noshell`. When the `noshell` executable is executed (as a login shell in `/etc/passwd`) a log entry is created and the shell exits. This allows administrators to track unauthorized use of system accounts.

cron and at Security

The cron and at systems execute commands at a future time. User submission for the cron system is handled by the crontab command. The at and batch commands are used to submit jobs to the at system.

Access to these commands can be restricted. The access control files are stored in the /usr/lib/cron directory. The cron.deny and cron.allow files manage access to the cron system. The at.deny and at.allow files are used to manage the access to the at system. The “allow” file is checked first to see if the account is explicitly allowed to use the system. If the file does not exist or the account is not listed in this file, the “deny” file is checked. If the account is explicitly listed in the “deny” file then access is refused. Otherwise, access is permitted. If neither the “deny” nor the “allow” files exist, then only the root account can use the at or cron system. The Solaris Operating Environment includes cron.deny and at.deny files which contain some system accounts.

The cron and at systems can be problematic since commands are executed at a future point in time. An attacker may use these systems to implement a “logic bomb” or other type of programmed attack that begins at some point in the future. Without examining every at and cron submission, tracking usage and abuse can be difficult.

It is better to restrict access to the at and cron systems to prevent attacks and abuse. By default, The Solaris Operating Environment includes scheduled cron events for the lp, adm, and root accounts. These should not be included in the “deny” files. Any additional system or software specific accounts that do not require cron or at access should be added to the “deny” files.

You may also want to restrict normal user access to these commands as well. Individual user accounts should be listed in the “deny” files. To restrict all user account access, create an empty “allow” file. Add only the accounts that need access to the “allow” file.

The init System

The Solaris Operating Environment init system manages system services. Some services may not be needed or should be modified to strengthen the security posture of a system.

System Default Umask

The default system file mode creation mask for the Solaris Operating Environment is 000. This means that files created by system daemons are created with permission bits that are 666 (readable and writable by all users). This can be a problem since normal users now have permission to overwrite the contents of system files. Use the following to set the system `umask` to a more reasonable value:

```
echo “umask 022” > /etc/init.d/umask.sh
chmod 744 /etc/init.d/umask.sh
chgrp sys /etc/init.d/umask.sh
for d in /etc/rc?.d; do
     ln /etc/init.d/umask.sh $d/S00umask.sh
done
```

The default system umask has changed to 022 in Solaris 8 Operating Environment. It can also be adjusted by altering the `CMASK` variable in the `/etc/default/init` file.

Disabling Services

System services are started by the `init` system. Some services are not necessary to system operation and should be disabled. There are also services that may allow a system to be compromised due to incorrect configuration. To disable services started by `init`, simply rename or delete the initialization script in the `init` system run level directory. The run level directories contain the scripts for starting or stopping services for the system run level. The system run level directories and their purpose is listed here:

/etc/rcS.d	single user
/etc/rc0.d	shutdown
/etc/rc1.d	start
/etc/rc2.d	multi-user
/etc/rc3.d	multi-user (default)
/etc/rc4.d	multi-user (unused)
/etc/rc5.d	shutdown and power off
/etc/rc6.d	shutdown and reboot

These directories contain initialization scripts to start or stop services. Initialization scripts that begin with either a “S” or a “K” are executed by the `init` system. “S” scripts start services, and “K” scripts stop or “kill” services. If you rename the scripts, make sure the name does not begin with these letters. We recommend placing and underscore (“_”) at the beginning of the name. This makes it easy to enable services that may be needed later. For example:

```
# cd /etc/rc.2
# mv S99dtlogin _S99dtlogin
```

For security purposes, only required services should be enabled. The fewer services that are enabled, the less likely it is that an attacker will discover a way to exploit the system using an enabled service.

The revision of the Solaris Operating Environment and the packages installed determine the services that are enabled by default. Removing unnecessary packages disables some extraneous services. The remaining services should be examined to determine their relevance to the system and the hosted application.

Kernel adjustments

There are several kernel adjustments that can be made to increase the Solaris Operating Environment security. The `/etc/system` file contains kernel specific parameter adjustments. Be careful when making changes to this file. Mistakes in this file may prevent the system from booting correctly.

NFS™ Server

By default, the Solaris Network File Service (NFS) server system accepts client NFS server requests from any port number. These requests should come from a privileged system port. The NFS server can be adjusted to only process requests from these privileged ports. If the system will act as an NFS server, add the following line to the `/etc/system` file for Solaris 2.5.1, 2.6, and 7 Operating Environment:

```
set nfssrv:nfs_portmon = 1
```

This change may prevent some NFS clients from operating correctly. There have been reported problems with older versions of Linux and SCO UNIX.

Executable Stacks

Some security exploitation programs take advantage of the Solaris Operating Environment kernel executable system stack to attack the system. These attack programs attempt to overwrite parts of the program stack of a privileged program in an attempt to control it. In Solaris 2.6 Operating Environment and later, some of these exploits can be avoided by making the system stack non-executable. Add the following lines to the `/etc/system` file:

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

With `noexec_user_stack_log` set to one, the system logs programmatic attempts to execute code on the stack. This allows you to track unsuccessful exploit programs and the account which made the attempt. Here is an example of a log message from a recent Solaris Operating Environment exploitation program that was stopped by enabling this feature:

```
Nov 28 11:59:54 landreth unix: sdtcm_convert[308] attempt to
execute code on stack by uid 38918
```

The problem with `sdtcm_convert` is corrected with a patch. However, the unpatched version of the program is somewhat resistant to the attack since the stack is not executable. Non-executable stacks provide some added protection against vulnerabilities for which no patch is issued.

This feature does not stop all buffer overflow exploitation programs, and it does not work on Intel x86-based or older SPARC hardware. Some overflow exploitation programs work on different principles which non-executable stacks cannot protect against. Always install the latest security patches. The non-executable stack feature only works on the following SPARC architectures: sun4d, sun4m, and sun4u hardware.

All 64-bit processes on Solaris 7 Operating Environment and later use non-executable stacks by default.

Core Files

Core files contain the memory image of an executing process which has received a certain signal and terminated. These files (with the file name `core`) are often used to investigate program errors. There are two problems with them. `core` files consume disk space and can contain sensitive information.

The size of the `core` file is based on the amount of memory consumed by the process during execution. A `core` file can take up a great amount of file space. A system with a full root ("/") file system may not perform as expected and may even crash.

In addition, the `core` file may contain privileged information that users should not be able to access. While running, the process may have read the `/etc/shadow` file to check a password or load a protected configuration file. These pieces of information are normally hidden from users but may exist in the process `core` file. This information may be used to attack the system. Add the following line to the `/etc/system` file to prevent the creation of `core` files:

```
set sys:coredumpsize = 0
```

For security reasons, the Solaris Operating Environment will not write `core` files for processes with an effective ID that is different from the real ID. This means that `set-user-ID` and `set-user-GID` programs will not create `core` files.

If `core` files must be used for application debugging, clean up old ones. From time to time, search the file system for old `core` files and delete them. This will help prevent the file system from becoming too full.

Solaris 7 Operating Environment, 8/99 and later releases include a new system utility for managing `core` files. The `coreadm` command allows an administrator to define directories and file name patterns for `core` files. It also allows `set-user-ID` programs to create `core` files for debugging purposes. The `set-user-ID` feature must be used with care and should be enabled only on development and testing systems. This feature can also be added to older Solaris 7 Operating Environment releases with patches 106541-06 (or later) for SPARC and 106542-06 (or later) for Intel systems. Solaris 8 Operating Environment includes it.

Log Files

Log file are used by the system and applications to record actions, errors, warnings, and problems. They are often quite useful for investigating system quirks, for discovering the root causes of tricky problems, and for watching attackers. There are typically two types of log files in the Solaris Operating Environment: system log files which are typically managed by the `syslog` daemon and application logs which are created by the application.

Log Files Managed by `syslog`

The `syslog` daemon receives log messages from several sources and directs them to the appropriate location based on the configured facility and priority. There is a programmer interface, `syslog()`, and a system command, `logger`, for creating log messages. The facility (or application type) and the priority are configured in the `/etc/syslog.conf` file to direct the log messages. The directed location can be a log file, a network host, specific users, or all users logged into the system.

By default, the Solaris Operating Environment defines two log files in the `/etc/syslog.conf` file. The `/var/adm/messages` log files contains a majority of the system messages. The `/var/log/syslog` file contains mail system messages. A third log file is defined but commented out by default. It logs important authentication log messages to the `/var/log/authlog` file. Uncomment the following line in `/etc/syslog.conf` to enable logging these messages:

```
#auth.notice ifdef(`LOGHOST', /var/log/authlog, @loghost)
```

Save the file and use the following command to force `syslogd` to re-read its configuration file:

```
# kill -HUP `cat /etc/syslog.pid`
```

All of these files should be examined regularly for errors, warnings, and signs of an attack. This task can be automated by using log analysis tools or a simple `grep` command.

Application Log Files

Application log files are created and maintained by commands and tools without using the `syslog` system. The Solaris Operating Environment includes several commands that maintain their own log files. Here is a list of some of the Solaris Operating Environment log files:

`/var/adm/sulog`	messages from `/usr/bin/su`
`/var/adm/vold.log`	messages from `/usr/sbin/vold`
`/var/adm/wtmpx`	user information from `/usr/bin/login`
`/var/cron/log`	messages from `/usr/sbin/cron`

The `/var/adm/wtmpx` file should be viewed with the `last` command.

The `/var/adm/loginlog` file does not exist in the default of the Solaris Operating Environment installation, but it should be created. If this file exists, the login program records failed login attempts.

All of these logs should also be monitored for problems.

Miscellaneous Configuration

The following configuration items apply to both local and remote security.

The `/etc/issue` File

The contents of the `/etc/issue` file are displayed on the console during login and for incoming telnet connections. It is often used to display information about the system or network. This file should contain warnings about inappropriate and unauthorized use of the system. It should also warn users that their sessions and accounts may be monitored for illegal or inappropriate use. Consult your legal counsel for more information.

Here is the legal warning found in the Titan security toolkit:

```
# This system is for the use of authorized users only.
# Individuals using this computer system without authority, or in
# excess of their authority, are subject to having all of their
# activities on this system monitored and recorded by system
# personnel.
#
# In the course of monitoring individuals improperly using this
# system, or in the course of system maintenance, the activities
# of authorized users may also be monitored.
#
# Anyone using this system expressly consents to such monitoring
# and is advised that if such monitoring reveals possible
# evidence of criminal activity, system personnel may provide the
# evidence of such monitoring to law enforcement officials.
```

The message of the day file (`/etc/motd`) can also be used to display warnings.

PAM

The Pluggable Authentication Module (PAM) architecture provides authentication, account management, session management, and password management mechanisms to applications in modular form. All the Solaris Operating Environment log in applications use the PAM system to authenticate users and manage accounts. Each PAM module can be implemented as a shared library object. The configuration file for the PAM system is `/etc/pam.conf`.

The PAM system exists to provide application programmers the ability to replace the methods used to manage accounts and users. For example, it may be desirable to limit the time periods that a group of users is allowed to be logged into a system. To implement this feature, a PAM module can be written to restrict users in this way without having to replace the log in programs.

To disable a specific log in method, remove or comment out its entry in the PAM configuration file. The `rlogin` and `rsh` services do not provide sufficient authentication for security and should be replaced with an SSH protocol system such as `ssh` (http://datafellows.com) or OpenSSH (http://openssh.com). Comment out the following lines in `/etc/pam.conf`:

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

If you disable the PAM configuration for `rlogin` and `rsh` services, also remove them from the `/etc/inet/inetd.conf` file. See the next section for more information.

Be careful when editing the `/etc/pam.conf` file. Errors will prevent all PAM services from operating and users will not be able to log in. To correct the problem, the system must be booted into single user mode. Also, do not change the original ownership or file permissions of the file. This will also prevent PAM from operating.

The `login` Command

The `login` command is part of the authentication process to access a local Solaris Operating Environment account. It is used on the console and by the `in.telnetd` daemon to determine if a user may be granted access to the system. By default, only the `root` user can log into a Solaris Operating Environment system from the console device. The console device is defined by the following entry in the `/etc/default/login` file:

```
CONSOLE=/dev/console
```

When this line is commented out, the `root` account log directly into the system over the network via `telnet` in addition to the console. This is insecure and should be avoided. Do not alter the default configuration.

Network Service Security

Network services provide client computer systems with additional features and applications normally unavailable to the average computer and user. They provide open access to information and data-driven services to many computers and users around the intranet and the Internet. However, this relatively open access must be tightly restricted and protected to prevent intrusion and misuse.

The Solaris Operating Environment is designed to provide customers with full access to most network services by default. This allows administrators and users to install and configure the Solaris Operating Environment systems as quickly as possible. Customers are encouraged to disable all unnecessary services for performance and security reasons.

Installation and minimization of the Solaris Operating Environment are important to the security of the system. This article discusses the network services provided when all Solaris Operating Environment bundled packages are installed (the *Entire Distribution* cluster). If a smaller installation cluster is used, then some of these services are not installed. The Solaris Operating Environment *Core* cluster contains the fewest number of packages and services. If the recommendations from the BluePrints Online article “Solaris Operating Environment Minimization for Security” are followed, then fewer network services are installed.

The network services a system provides are the entry points into that system. It is important to know the default for the Solaris Operating Environment services, and the methods used to disable them. Often, organizations must use protocols or services that are not secure. For commonly used services (such as RPC, NFS, and Trivial FTP), suggestions are given for how to improve security.

Services offered by a system should be protected by as many layers of security as possible. This protection should start at the network level. The December 1999 issue of Sun BluePrints Online included an article entitled “Solaris Operating Environment Network Settings for Security”. It discussed and recommended alternative parameter settings for low-level network protocols such as IP, ARP, TCP, and ICMP.

Network Service Issues

There are many possible ways to attack network services. These services contain programming flaws, use weak authentication, transfer sensitive data in unencrypted format, and allow connections from any network host. These weaknesses allow a system to be compromised by an attacker.

There are some quick and easy methods to prevent successful attacks. Administrators should always disable unneeded services and apply all security patches. In addition, safer network service alternatives should be used whenever possible. Sun has a product for secure network services, based on Kerberos, called the Sun Enterprise Authentication Mechanism™ (SEAM product). Kerberos is a centralized network security architecture that uses a ticket mechanism to provide strong authentication. The SEAM product also uses strong encryption. There is also a tool known as Secure Shell (SSH) that provides strong authentication and encryption capabilities. There are both commercial and open source versions available. Access control can be provided, thereby configuring network services to only handle connections from approved systems. Wietse Venema's TCP wrapper toolkit provides access control and additional security checks. It manages TCP-based services managed from `inetd`.

Telnet

Telnet is a user interactive service used to log into and access a remote system on the network. Unfortunately, we believe this service provides little in the way of security. The only authentication information required is user name and password. Neither of these pieces of information are encrypted while in transit and are therefore vulnerable to a variety of attacks including: man in the middle attack, session hijacking, and network sniffing. The SEAM product provides a replacement `telnet` command that uses strong authentication and encryption. SSH also serves as an effective replacement.

If you must use a `telnet` daemon which does not support encryption, then One Time Passwords and TCPWrappers should be used to secure the connections. One Time Passwords (OTP) protect against network sniffing by never transmitting the password over the network. Instead, a challenge issued by the server in combination with a secret phrase is used to generate the password used for authentication. TCPWrappers can be used to limit the hosts which may connect to a system. By restricting access to services based on IP addresses, a system can limit its exposure to network attacks.

Remote Access Services (rsh, rlogin, and rcp)

Access control and accountability are critical to the security of a system. Access control should involve strong authentication for system access, while accountability information provides tracking information relative to system changes. The standard r* commands (i.e., rsh, rlogin, and rcp) break both of these requirements. This is because most implementations of r* commands involve “zones of trust”. Within a zone of trust, all systems are trusted and no additional authentication is required. Hence, an intruder need only gain access to one server in order to gain access to all the servers.

The default authentication mechanism of the r* daemons uses the IP address of a system in combination with the userid for authentication. No additional authentication is required. Considering the ease with which an IP address and userid may be stolen or misused, this is clearly not a secure mechanism. The r* commands should never be used in this manner and no servers should offer the service in this manner.

Secure Shell (SSH) can be used to improve the security of the r* commands by:

- encrypting network traffic between client and server
- requiring RSA authentication through key pairs
- utilizing manual key exchange
- allowing only known hosts to communicate

Another manner in which the r* daemons may be secured is with Kerberos. The SEAM product provides the appropriate replacement for r* clients and servers.

Remote Execution Service (rexec)

The remote execution server daemon, in.rexecd, is started from /etc/inetd.conf when a connection request is made. This daemon provides remote execution facilities based on user name and password information. Once authenticated, the daemon executes the command passed along with the authentication information. As with the in.telnetd daemon, neither the user name nor password is encrypted while transmitted over the network. This exposes the in.rexecd daemon to the same man in the middle, session hijacking, and network sniffing attacks as the in.telnetd and in.ftpd daemons. For this reason the in.rexecd entries in /etc/inetd.conf should be removed.

FTP

The FTP daemon has many of the same problems as the `telnet` daemon. All authentication information transmitted over the network is in clear-text, in much the same fashion as the `telnet` protocol. This exposes the `ftp` protocol to many of the same attack scenarios as `telnet`, including man in the middle, session hijacking, and network sniffing. For these reasons, alternatives to FTP should be considered when FTP transport functionality is required.

There are several alternatives to FTP which provide strong encryption and authentication. Two of the most popular are Secure Shell (SSH) and Kerberized FTP. SSH is designed as a drop-in replacement for the Remote Copy command (`rcp`) but it can also function as a replacement for FTP. A true replacement for FTP is the Kerberos-compliant FTP client/server distributed as part of the SEAM product.

If you must use FTP, there are two features implemented by the `in.ftpd` daemon which can provide additional security. The first is the `/etc/ftpusers` file which is used to restrict access to the system through FTP. All accounts *not* allowed to use the incoming FTP service should be specified in this file. At a minimum, this should include all system accounts (i.e., `bin`, `uucp`, `smtp`, `sys`, and so forth) in addition to the `root` account. Only intruders and those individuals attempting to gain unauthorized access use these accounts. Frequently, `root` access to a server over `telnet` is disabled but `root` FTP access is not. This provides a backdoor for intruders which may be used to modify the systems configuration by uploading modified configuration files.

The second security feature of the `in.ftpd` daemon is the ability of the daemon to log the IP addresses of all connections to the `ftp` daemon through the `syslog` service. This is enabled with the `-l` option. By logging FTP connection requests and forwarding them to a log server for parsing, unauthorized access attempts can be tracked and resolved.

Trivial FTP

The trivial FTP service (`in.tftpd`) exists to provide disk-less systems with a way to get files on the network. The `in.tftpd` daemon does not require a user name and password for access. Due to this lack of authentication, `in.tftpd` allows only publicly readable files to be read and updated. Disk-less workstations, x-terminals, and some printers use this service to load files needed to boot. `in.tftpd` is managed by the `inetd` server process and is configured in `/etc/inetd.conf`. By default, it is not enabled in the Solaris Operating environment.

If this service is necessary, it should be configured for security. The `in.ftpd` daemon does provide system administrators the ability to restrict the directory which is shared through the `in.tftpd` daemon. This option forces the `in.ftpd` daemon to change into the specified directory and only share publicly readable

information contained in that directory. By default, the standard Solaris Operating Environment `/etc/inetd.conf` file, has this option enabled and set to `/tftpboot`. The “-s” option is used to specify the appropriate directory. This configuration option should always be used when TFTP functionality is required.

`inetd` Managed Services

`inetd` manages a majority of the minor network services available on a system. It’s configuration file, `/etc/inetd.conf`, defines its operation. An ideal secured server should neither have an `/etc/inetd.conf` nor run `inetd`, as the daemons started in the `/etc/inetd.conf` are frequently not needed. To disable a service, edit the `/etc/inetd.conf` file and place a comment character (“#”) in front of the line containing the service definition. Once this is completed, send a HUP signal to the `inetd` process. This will cause it to reread its configuration file.

Of the daemons started from the `/etc/inetd.conf`, the remote access services, FTP, TFTP, and TELNET services have already been discussed. The RPC and print services are discussed a little later. The remaining `/etc/inetd.conf` entries include:

- `in.tnamed`—a server that supports the DARPA Name Server Protocol. This daemon should be disabled.
- `in.uucpd`—a server that supports UUCP connections over networks. This service should be disabled unless UUCP is used.
- `in.fingerd`—a service that provides information on local system accounts. This service should be disabled unless needed.
- `systat`—a service that provides anyone connecting to the system with the output of `ps -ef`. This service should be disabled because it provides too much system information.
- `netstat`—a service that provides a list of current network connections via the output of the `netstat` command.This service should be disabled because it provides too much system information.
- `time`—a service that prints out the current time and date. With Solaris 2.6 Operating Environment `xntp` functionality has been included with the Solaris Operating Environment distribution for time synchronization. The `xntp` daemon offers additional security and functionality improvements over `rdate` and `time`. Whenever possible `xntp` should be used instead of this service.
- `echo`—a service that echoes back the incoming data stream. This service should be disabled.
- `discard`—a service that discards the incoming data stream. This service should be disabled.
- `chargen`—a service that generates a continuous stream of characters. This service should be disabled.

These entries in the `/etc/inetd.conf` file should be removed on most systems. Once removed, the server should be restarted and applications tested to verify that required functionality has not been affected.

For restricted access servers, it may be helpful to have a log of all requested connections to services being started out of the `/etc/inetd.conf` file. This can be done by adding an additional option to the startup of `inetd` in `/etc/rc2.d/S72inetsvc`. By adding a `-t` option, the `inetd` daemon logs the IP address of all systems requesting `inetd` based services. The IP addresses are logged through the `syslog` service.

RPC Services

Remote Procedure Call (RPC) services are used in many UNIX services including: NFS, NIS, NIS+, and Kerberos. RPC services are also used by many applications such as Solstice Disk Suite™ software, SunCluster software, and others. All of these daemons and applications use the `rpcbind` daemon for converting RPC program numbers into universal addresses.

When an RPC service is started, the service tells the `rpcbind` daemon the address where it is listening and the RPC program numbers it is prepared to serve. When a client wishes to make an RPC call to a given program number, it first contacts the `rpcbind` daemon on the server machine to determine the address where RPC requests should be sent. The `rpcinfo` command can be used to determine what RPC services are registered on a host.

RPC, by itself, can be used to provide an attacker with information about a system. While this may not be ideal, the real security problem is not the `rpcbind` daemon itself, but rather many of the services that use RPC. Many of these services do not make use of the stronger authentication mechanisms available to them and default to weak authentication. In particular, `rpc.cmsd`, `sadmind` (running without `-S 2`), and `rpc.rexd` use weak authentication by default. Network based attacks against these services pose a significant threat to the security of a server.

The daemons and services which use RPC on a Solaris Operating Environment system include:

From `/etc/inetd.conf`:

- `testsvc`
- `sadmind`
- `rquotad`
- `rpc.rusersd`
- `rpc.sprayd`
- `rpc.rwalld`

- rpc.rstatd
- rpc.rexd
- ufsd
- kcms.server
- fs
- cachefsd
- kerbd
- in.lpd
- dtspcd
- xaudio
- rpc.cmsd
- rpc.ttdbserver

From /etc/rc2.d/S71rpc:

- rpcbind
- keyserv
- rpc.nisd
- nis_cachemgr
- rpc.nispasswdd

From /etc/rc3.d/S15nfs.server:

- rpc.bootparamd

On almost all servers, the RPC services in /etc/inetd.conf can be removed. Many applications which use RPC services will add additional entries to the /etc/inetd.conf in addition to using one of the RPC based daemons. The RPC services in /etc/inetd.conf should be removed unless specifically required.

The RPC daemons started in /etc/rc2.d and /etc/rc3.d are for rpcbind, keyserv, various naming services (i.e., NIS and NIS+), and are also used by both the client and server components of NFS. The keyserv daemon must be run when AUTH_DES is used for stronger host and user authentication. The use of NIS is not recommended due to its weak encryption and authentication models. NIS+ provides a much more robust security model.

The RPC protocol provides support for various authentication alternatives. These include:

- AUTH_NONE—No authentication.
- AUTH_SYS or AUTH_UNIX—Traditional UNIX-style authentication.
- AUTH_DES—DES encryption-based authentication.
- AUTH_KERB—Kerberos encryption-based authentication.

Some RPC daemons and services provide options for an administrator to specify the security model (e.g., NFS, `sadmind`, NIS+) while others do not. If RPC must be used, then only those services and daemons which provide support for AUTH_DES should be used. This combination of RPC and AUTH_DES authentication is called Secure RPC. See “Bibliography” on page 32 for additional references to Secure RPC.

NFS™ Server

A Solaris Operating Environment system can be either an NFS server, NFS client, both, or neither. From a security perspective, the best option is to neither provide NFS services nor accept them from any other systems. To disable all client and server NFS daemons the following startup scripts should be disabled on the system:

- `/etc/rc1.d/K65nfs.server`
- `/etc/rc1.d/K80nfs.client`
- `/etc/rc2.d/S73nfs.client`
- `/etc/rc2.d/K60nfs.server`
- `/etc/rc3.d/S15nfs.server`

The Solaris Operating Environment uses a different set of startup files to enable NFS server or NFS client services.

Frequently, business requirements mandate the use of the NFS server. There are several different levels of security available in the NFS server itself. In addition, careful configuration can also greatly improve security. Here is a quick overview:

- Explicitly list hosts allowed access to NFS server directories. Do not open access to all systems.
- Export only the lowest directory necessary.
- Export read-only whenever possible.
- Use strong authentication methods such as AUTH_DES or AUTH_KERB whenever possible.

The NFS server and the various mechanisms available to secure it encompass more material then can be discussed here. A future BluePrints Online article will be dedicated to a detailed discussion of the NFS server and various authentication and encryption mechanisms for it.

Automount

The automount service manages automated NFS mounts. The `automount` utility installs `autofs` mount points and associates an `automount` map with each mount point. The `autofs` kernel module monitors attempts to access these mount points

and notifies the automountd daemon. The daemon uses the map to locate a file system, which it then mounts at the point of reference within the autofs file system. You can assign a map to an autofs mount using an entry in the /etc/auto_master map or a direct map.

In Solaris 2.6 and 7 Operating Environment, the automount software is packaged separately. By removing these packages, all automount functionality is removed from the system. The two packages which include all the automounter functionality are SUNWatfsr and SUNWatfsu.

The file /etc/auto_master determines the locations of all autofs mount points. By default, this file contains four entries:

```
 # Master map for automounter
#
+auto_master
/net -hosts -nosuid
/home auto_home
/xfn -xfn
```

Ideally, it should be disabled since, not only does it run as a privileged daemon, but it also uses NFS and RPC. The automounter can be disabled by renaming the /etc/rc2.d/S74autofs file.

There are situations where the automount service is needed for its ability to mount and unmount file systems automatically. In particular, both NIS and NIS+ environments make extensive use of auto_home and auto_master maps to mount user home directories. In these situations, the configuration of the auto_master map should be carefully constructed to be as restrictive as possible. This can be done by using the standard NFS mount options.

Sendmail

sendmail is used on a Solaris Operating Environment system both to forward and receive mail from other systems. Centralized mail servers should be used to receive email and not local servers. These local systems should, however, be able to generate email and forward it to other servers.

Ideally, a more secure Mail Transport Agent (MTA) should be used instead of the MTA bundled with the Solaris Operating Environment. The sendmail daemon, bundled with the Solaris Operating Environment, has been subject to numerous denial of service, buffer overflow, and misconfiguration attacks. Alternative MTAs have been developed with smaller and more robust code. These other MTAs are

more security conscious and, if configured properly, compromise the security of the server less than `sendmail`. If `sendmail` must be used, then the following recommendations should be followed to secure it as much as possible.

Disable `sendmail` Daemon

The first step in disabling `sendmail`, in Solaris 2.6 Operating Environment and earlier Solaris Operating Environment releases, is to rename the `sendmail` startup scripts. The `/etc/rc2.d/S88sendmail` and `/etc/rc1.d/K57sendmail` scripts should be renamed to disable the mail service. On Solaris 7 Operating Environment systems, it is possible to remove all components of `sendmail` by removing the `SUNWsndmr` and `SUNWsndmu` packages with `pkgrm`.

Outgoing Sendmail

The `sendmail` daemon is not needed for E-mail delivery to other systems. All messages that can be immediately delivered, are. Messages that cannot be immediately delivered are queued for future delivery. The `sendmail` daemon, if running, retries these messages again. It is recommended that a `cron` job be used to start `sendmail` every hour to process these undelivered messages. The following cron entry starts `sendmail` every hour to flush the mail queue:

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

sendmail.cf Recommendations

There is a wide variety of `sendmail` versions in use, and there are differences in the associated `sendmail.cf` configuration files. Because of this, a sample `sendmail.cf` file is not included with this article. Please refer to recommendations made at Sendmail Consortium, in the Sendmail O'Reilly books, and through the SunSolve OnLine^{SM} program .

Name Service Caching (`nscd`)

The `nscd(1M)` daemon provides a cache for the most common name service requests. It is started up during multi-user boot. The default configuration file, `/etc/nscd.conf`, determines the behavior of the cache daemon. See the `nscd.conf` man page for more information.

nscd provides caching for the passwd, group, and hosts databases through standard libc interfaces, such as gethostbyname, gethostbyaddr, and others. Each cache has a separate time-to-live for its data and modifying the local database (/etc/hosts, and so forth) invalidates that cache within ten seconds. One file which is specifically not cached is the /etc/shadow file as it contains sensitive information which should never be cached for security reasons.

The -g option to nscd can be used to view the current configuration of nscd on a server.

When not specifically required, it is suggested that nscd be disabled entirely by commenting out the /etc/rc2.d/S76nscd file. If nscd functionality is required by the server, because it is running NIS or NIS+ for instance, then the /etc/nscd.conf file should be configured to cache as little information as possible. In particular, the configuration should be modified so that neither passwd nor group information is cached. Keep in mind that disabling this service or reducing its caching ability may impact performance with systems that have many users.

A sample configuration file for /etc/nscd.conf with passwd and group caching disabled is shown below:

```
enable-cache            passwd          no
enable-cache            group           no
positive-time-to-live   hosts           3600
negative-time-to-live   hosts           5
suggested-size          hosts           211
keep-hot-count          hosts           20
old-data-ok             hosts           no
check-files             hosts           yes
```

Print Services

When a Solaris Operating Environment system is installed using the *End User*, *Development*, or *Entire Distribution* cluster, the line printing packages are installed. Both the client and server components for print services are enabled by default on these Solaris Operating Environment installations.

The in.lpd daemon is only necessary for systems providing network based print services to other servers. If this is not the case, then the following line should be removed from the /etc/inetd.conf file:

```
printer stream tcp nowait root /usr/lib/print/in.lpd in.lpd
```

Conversely, the `/etc/rc2.d/S80lp` script is required both for a server providing print services to other systems and a system which requires access to printers hosted by other systems. If this functionality is not required, the packages for `lp` should be removed from the system, and the `in.lpd` entry should be removed from `/etc/inetd.conf`. The three packages for `lp` are `SUNWpsr`, `SUNWpsu`, and `SUNWlpmsg`.

IP Forwarding

During the startup phase of a Solaris Operating Environment system, the `/etc/init.d/inetinit` script evaluates the configuration of the system. It determines whether or not the system will be configured as a router and have `ip_forwarding` enabled between the different interfaces. For more information on the `ip_forwarding` function, refer to the BluePrints Online article entitled “Solaris Operating Environment Network Settings for Security”.

Network Routing

The router (`in.routed`) and router discovery (`in.rdisc`) daemons are used by a Solaris Operating Environment system to dynamically determine the routing requirements of networks. Both the `routed` and `rdisc` functionality have been discussed in a previous BluePrints Online article entitled “Solaris Operating Environment Network Settings for Security”.

Multicast Routing

Unless the server is participating in the MBONE multicast network application, the `multicast` startup script can be commented out in the `/etc/init.d/inetsvc`. The removal of multicast support is recommended unless it is specifically needed by an application or to satisfy a business requirement.

To disable multicast support, the following lines should be commented out of the `/etc/init.d/inetsvc` file:

```
mcastif=`/sbin/dhcpinfo Yiaddr`
if [ $? -ne 0 ]; then
        mcastif=`uname -n`
fi
echo “Setting default interface for multicast: \c”
/usr/sbin/route add -interface -netmask “240.0.0.0” “224.0.0.0”
“$mcastif”
```

Once these lines are commented out, the system should be restarted.

Reducing `inetsvc`

Based on the recommendations made in this article, it is possible to construct a minimized `/etc/init.d/inetsvc` file which contains only the essential components. Quite a few sections of this file can be commented out including:

- DHCP support
- named startup support
- multicast support

By commenting out all of these entries, the number of active lines in the `inetsvc` file decreases from 152 to 3 lines. The following is what the resulting script looks like:

```
#!/bin/sh

/usr/sbin/ifconfig -au netmask + broadcast +
/usr/sbin/inetd -s -t
```

Network Service Banners

Some Solaris Operating Environment network services provide information on the operating system version when connections are made. It usually includes a text string indicating the name of the OS and its version. This information maybe useful to attackers with exploit programs for specific OS releases. The Solaris Operating Environment provides a method to change these messages in an attempt to hide OS information.

To change banner messages for incoming telnet and FTP connections create the `/etc/default/telnetd` and `/etc/default/ftpd` files. Add a line similar to the following:

```
BANNER="Unlisted OS"
```

Insert the appropriate message for your environment.

It is also possible to change the banner message that the `sendmail` process presents for incoming mail delivery connections. Search the `/etc/mail/sendmail.cf` file for the following line:

```
O SmtpGreetingMessage=$j Sendmail $v/$Z; $b
```

Change it to:

```
O SmtpGreetingMessage=Mail Server Ready
```

These techniques provide only minor additional security. There are methods to determine a system's operating system type and version on a network. Several network auditing tools use a technique called "TCP/IP stack fingerprinting" to determine the operating system and version.

Summary

Securing a Solaris Operating Environment system requires that changes be made to its configuration. The changes outlined in this article address the majority of the methods used to gain unauthorized or privileged access to an improperly configured system. The implementation of these changes require planning, testing, and documentation in order to be successful in securing a computing environment.

Bibliography

AUSCERT, *UNIX Security Checklist*,
ftp://ftp.auscert.org.au/pub/auscert/papers/unix_security_checklist

Hal Pomeranz, *Solaris Security Step by Step*, http://www.sans.org/

Jason Rhoads, *Solaris Security Guide*, http://www.sabernet.net/papers/Solaris.html

Lance Spitzner, *Armoring Solaris*, http://www.enteract.com/~lspitz/armoring.html

Peter Baer Galvin, *The Solaris Security FAQ*,
http://www.sunworld.com/common/security-faq.html

Sendmail Consortium, sendmail configuration information,
http://www.sendmail.org/

Alex Noordergraaf, Keith Watson, *Solaris Operating Environment Minimization for Security*, http://www.sun.com/blueprints/1299/minimization.html

Alex Noordergraaf, Keith Watson, *Solaris Operating Environment Network Settings for Security*, http://www.sun.com/blueprints/1299/network.html

Brad Powell, et al., Titan security tool, http://www.fish.com/titan/

Wietse Venema, TCP Wrappers tool,
ftp://ftp.porcupine.org/pub/security/index.html

Casper Dik, `fix-modes` tool, ftp://ftp.wins.uva.nl/pub/solaris/fix-modes.tar.gz

SSH Communications Security, Secure Shell (SSH) tool, http://www.ssh.org/

Sun Enterprise Authentication Mechanism (SEAM),
http://www.sun.com/solaris/ds/ds-seamss/

Author's Bio: Alex Noordergraaf

Alexander Noordergraaf has over eight years of experience in the area of Computer and Network Security. As a Senior Security Architect for SunPS Global Enterprise Security Service (GESS), he has worked with many Fortune 500 companies on projects that include Security Assessments, Architecture Development, Architectural Reviews, and Policy/Procedure review and development. His customers have included major telecommunication firms, financial institutions, ISPs, and APSs.

Author's Bio: Keith Watson

Keith Watson has spent the past two years at Sun developing an enterprise network security auditing tool suite named the Sun Enterprise? Network Security Service (http://www.sun.com/software/communitysource/senss/). He currently works for the SunPS Global Enterprise Security Service (GESS) consulting practice. Prior to joining Sun, he was part of the Computer Operations, Audit, and Security Technologies (COAST) laboratory at Purdue University.