Chapter 4

System Security

This chapter deals with maintaining the security of your local computer system. Once you have initially established the security of a system, you can expand your secure area to include the network. But until you have local security, there is no point in trying to establish security over a larger area.

In addition, security is never established *finally*. That is, security is a dynamic process, requiring that you understand the issues, keep up to date on them, and continually monitor your system with the many tools available. It is the intention of this chapter and the next to give you the information you need to establish a security policy and begin its implementation.

Chapter 5 discusses *network* security issues, and is intended for use after the issues discussed in this chapter have been addressed to your satisfaction. If you are not connecting to the Internet or any large network scheme but only establishing or connecting to a small, local area network, you should still read the first part of Chapter 5, "Local Area Network Access" on page 107.

This chapter includes the following sections:

- "Standard Security Features" on page 80, which is an overview of the standard security features incorporated in IRIX design.
- "Security Guidelines" on page 81 is a list of areas to check for common security holes.
- "Password Administration" on page 84 details proper setup and control of system software and user accounts to increase security.
- "Login and Account Administration" on page 93 covers proper maintenance of special accounts as well as user logins.
- "Set-UID and Set-GID Permissions" on page 99 describes the nature and control of the file permissions that enable user and group IDs to be set on execution.
- "General File and Directory Permissions" on page 102 discusses permission settings and lists the IRIX files and directories that are universally available for read and write access.

- "Accounts Shipped Without Passwords" on page 103 lists the user accounts in */etc/passwd* that do not contain passwords on the software as shipped.

A great strength of the IRIX system is the ease with which users can share files and data. However, some of the security guidelines presented in this chapter are at odds with easy system access. It is up to you as the system administrator to balance the needs and concerns of the user community.

It is one thing to secure an isolated IRIX system, another to secure a local area network, and still another to secure a site that is connected to external networks such as the Internet. This chapter deals primarily with taking steps to secure an isolated system, but many of these same steps must also be taken before undertaking the more ambitious job of securing a network. For information regarding network security issues, refer to Chapter 5.

Standard Security Features

IRIX has several features that allow you to achieve a generally acceptable level of security without adding any new software. Standard security features of IRIX are:

- file ownership split into three classes—*owner*, *group*, and *other*—permits the owner of files to specify who is allowed access to the files
- permissions split into three categories—*read*, *write*, and *execute*—permits the owner of files to specify the degree of access users may have to the files
- the ability to encrypt data, using the crypt(1) command
- individual user accounts, protected by individual, encrypted passwords
- tools for monitoring login attempts
- tools for monitoring system activity, including:
  - finding out which processes are running, using the ps(1) command
  - determining who is logged on the system, using the who(1) command
  - maintaining logs of system activity, using process accounting commands

Security Guidelines

Computer security is the responsibility of not only the site administrator, but of everyone who has access to a computer at the site.

System users should safeguard their data by using appropriate file and directory permissions in addition to using and guarding their account passwords.

Site administrators, and to some extent system users, should be aware of the following:

- Anyone with physical access to a computer can simply take it or take its disk drives(s).
- The same caveat applies to backups of the system: keep backups in a secure place. Anyone with physical access to backup tapes can gain access to any information stored on them.
- Permissions for directories and files should be set to allow only the necessary access for owner, group, and others. This minimizes the damage that one compromised account can cause.
- There are several ways accounts and passwords protect the system:
  - By requiring users to log in with specific accounts, you can determine who is responsible for specific actions on the system.
  - Using the IRIX system of file permissions, users can keep data reasonably secure. Other users on the system are less likely to accidentally view confidential material.
  - If all accounts have passwords, the chance of an unauthorized person accessing the system is greatly reduced. However, the possibility of unauthorized access increases if users are lax about changing their passwords regularly and choosing good passwords. The next section describes how to choose good passwords.
- All active accounts need passwords, which should be changed regularly. Do not use obvious passwords, and do not store them online in “plain-text” format. If you must write them down on paper, store them in a safe place.

  For information about choosing passwords, see “Choosing Passwords” on page 84.
- Common-use accounts are a potential security hole. An example of a common-use account is one that is shared by all members of a department or work group. Another example is a standard “guest” account on all the workstations at a site.

This allows all users at the site access to different workstations without requiring specific accounts on each workstation.

A pitfall of common-use accounts is that you cannot tell exactly who is responsible for the actions of the account on any given workstation. Another risk is that anyone trying to break into workstations at your site will try obvious account names such as *guest*.

Common-use accounts can be helpful, but be aware that they can pose serious security problems. Needless to say, common-use accounts that do not have passwords are especially risky.

- Accounts that are no longer used should be either locked or backed up and removed, since unused accounts can be compromised as easily as current accounts.

  Also, change critical passwords, including dialup passwords, whenever anyone leaves the organization. Former employees should not have access to workstations at the site.

- Systems with dialup ports should have special dialup accounts and passwords. This is very important for sites that have common-use accounts, as discussed above. Refer to the discussion on */etc/d_passwd* in “Second (Dialup) Passwords” on page 87.

  However, even with this added precaution, do not store sensitive data on workstations that have dial-up access.

- If your site allows access to the Internet network (for example, using ftp(1C)), take precautions to isolate access to a specific gateway workstation. Refer to “Network Security and Firewalls” on page 111 for details on connecting to external networks.

- Discourage use of the su(1) command unless absolutely necessary. The *su* command allows a user to change his or her user ID to that of another user. It is sometimes legitimately necessary to use *su* to access information owned by another user, but this presents an obvious temptation: the person using *su* to switch user IDs must know another person’s password and therefore has full access to his or her account.

  Note: The file */var/adm/sulog* contains a log of all successful and unsuccessful attempts to use the *su* command (if it is enabled in */etc/default su*).

- Make sure that each user’s home account, and especially the shell-startup files *.profile*, or *.login* and *.cshrc,* are writable only by that user. This ensures that “trojan horse” programs are not inserted in a user’s login files. (A trojan horse program is a file that appears to be a normal file, but in fact causes damage when invoked by a legitimate user.)

- Be sure that system directories such as / (root), */bin*, */usr/bin*, and */etc* and the files in them are not writable except by the owner. This also prevents trojan horse attacks.
- If you must leave your console, workstation, or terminal unattended, log off the system. This is especially important if you are logged in as *root*. Also, refer to the xlock(1) reference page for information on locking your local X display.
- Sensitive data files should be encrypted. The crypt(1) command, together with the encryption capabilities of the editors (*ed* and *vi*), provides some protection for sensitive information.
- Use only that software that is provided by reputable manufacturers. Be wary of programs that are distributed "publicly," especially already-compiled binaries. Programs that are available on public bulletin board systems (as opposed to BBSs run and sponsored by vendors) and on public computer networks could contain malicious "worm" routines that can violate system security and cause data loss.

  Public-domain source code is safer than already-compiled programs, but only if you examine the code thoroughly before compiling it. Be suspicious of programs that must be installed with set-UID *root* in order to run.
- Safeguard and regularly check your network hardware. One possible way to break into computer systems is to eavesdrop on network traffic using physical taps on the network cable. Taps can be physical connections (such as a vampire tap) or inductive taps.

  Run networking cable through secure areas and make sure it is easy to examine regularly. Create and maintain a hard copy map of the network to make it easier to spot unauthorized taps. Another way to make this sort of attack less likely is to use fiber-optic (FDDI) network hardware, which is much more difficult to tap. For details on configuring network software securely, refer to Chapter 5.

System security under IRIX is primarily dependent on system login accounts and passwords. Proper administration, user education, and use of the facilities provided yield adequate security for most sites. Most security breaches are the result of human error and improper use of the provided security features. *No extra measures yield more security if the basic features are not used or are compromised by user actions*. Also, periodically log in with anonymous FTP to *sgigate.sgi.com* and look in the directory *~ftp/security* for any security patches for your system.

Note: If you are using NFS or NIS on your system, see the discussions in "Disabling NIS (YP)" on page 124 and "Limiting NFS Access" on page 125.

Password Administration

This discussion of password administration includes the following sections:

- "Choosing Passwords" on page 84 discusses how to choose more secure passwords and avoid easily guessed ones.
- "PROM Passwords" on page 85 discusses use of PROM passwords.
- "Second (Dialup) Passwords" on page 87 discusses how to associate a second password (often called system or dialup passwords) with specific tty lines.
- "Creating a Shadow Password File" on page 89 describes the use of a shadow password file to hide even the encrypted passwords contained in the standard password file.
- "Password Aging" on page 90 shows how to force users to choose new passwords at specified intervals.
- "Using pwck to Check the Password File" on page 93 introduces a useful tool that performs some useful password file checks.

Managing passwords is also described in "IRIX Admin: System Configuration and Operation."

Choosing Passwords

A system is most secure if nobody can access the system without an account and password, and if all the passwords on the system are difficult to guess and obtain. Surprisingly, many users choose passwords that are easy for potential intruders to guess, or write their passwords down on paper and leave them near their workstations and terminals.

Also, many site administrators use the same password for multiple administrative accounts. This is not a good practice. Do not deliberately use the same password for more than one account.

More secure passwords are:

- long (the first eight characters are recognized)
- multiple words that are combined or arranged in an unusual manner
- words from multiple languages, combined in a unique way

- composed of different kinds of characters, such as digits and punctuation
- have all of these bulleted features

Easily guessed passwords are:

- short
- single words that are in a dictionary
- the same as the account name, or the account name spelled backward
- the name of the user's department or project
- the user's name or initials
- the license number of the user's car, a spouse or friend's name, the user's home address, phone number, age, or other obvious information
- obvious—for example, "top secret," "secret," "private," "password," "friend," "key," "god," "me," and so on

PROM Passwords

Your system has a facility that allows you to require a password from users who attempt to gain access to the Command (PROM) Monitor. This gives you greater control over who may perform system administration tasks.

Traditionally, if an intruder gains access to your system hardware, there is little you can do to maintain system security. In the simplest case, the intruder switches off the system, then turns it on again, and instructs the system from the console to boot a program other than your usual operating system. Alternatively, the intruder could simply remove the hard disk from your system and install it on another system and read your files. While there is nothing you can do with system software to prevent physical theft of the hardware, you can limit the ability of intruders to boot their programs or to otherwise damage your system at its lowest levels with a PROM password.

Note that if you forget your PROM password, but you still know your *root* password, you can reset the PROM password on most systems through the *nvram* command. If you cannot successfully reset the PROM password, you must remove the PROM or a jumper from your CPU board. See your *Owner's Guide* for information on this procedure.

To assign a new PROM password if you have forgotten it, first clear the existing PROM password from IRIX with the *nvram* command, and then assign a new one with the *passwd* command from the PROM monitor.

Clearing the PROM Password Using nvram

To clear the PROM password using the nvram(1M) command, perform the following steps:

1. Log in as *root*.
2. Give the following command:

   `nvram passwd_key ""`

   Your PROM password is now cleared.

Setting the PROM Password From the Command Monitor

If you wish to set your PROM password from within the Command Monitor, perform the following steps:

1. Log in as *root* and shut the system down.
2. When you see the message:

   ```
   Starting up the system...
   ```

   ```
   To perform system maintenance instead, press <Esc>
   ```

   press the **`<Esc>`** key to see the System Maintenance Menu.
3. Select option 5 from the System Maintenance Menu to enter the Command Monitor. You see the Command Monitor prompt:

   ```
   >>
   ```
4. Type the *passwd* command and press **`<Enter>`**:

   ```
   passwd
   ```

   You see the prompt:

   ```
   Enter new password:
   ```
5. Enter the password you want for your system and press **`<Enter>`**. You see the following prompt:

   ```
   Confirm new password:
   ```

6. Enter the password again, exactly as you typed it before. If you typed the password the same as the first time, you see the Command Monitor prompt again.Your password is now set. Whenever you access the Command Monitor, you will be required to enter this password.

Refer to "Choosing Passwords" on page 84 for help in selecting a good password.

Second (Dialup) Passwords

If your system requires additional protection, you can establish a *system password*. If you do this, users who log in on specific ports (ttys) are prompted for a system password in addition to their account passwords. This feature cannot be imposed on the system console, or any terminal where *clogin* or *xdm* is used.

System passwords are normally used only on dialup lines and are often referred to as *dialup passwords*. You can use them on standard lines, but this is usually not necessary.

To establish a system password, follow these steps:

1. Log in as *root*.
2. Edit the file */etc/dialups*.

   Place in the file a list of ports (ttys) that require the second password. For example:

   ```
   /dev/ttyd1
   /dev/ttyd2
   /dev/ttyd3
   ```

   All possible names for ports should be listed including links. Write the file and exit from the editor.
3. Decide on the desired password or passwords. System passwords are assigned on a shell-by-shell basis. You can assign the same password for all the possible shells on the system, assign different passwords for each shell, or use some combination of approaches.
4. Encrypt the desired password. You must use the *passwd* program to perform the encryption. You cannot use the crypt(1) command for this purpose.

   To encrypt the password, simply change the password of some account (for example the *bin* account) to the password you wish to use in */etc/d_passwd*. Before you do this, note what the existing password is (or if the account is locked). Return the account to this state when you are finished assigning a system password. (To save an account's existing password, copy the password field of the account—the second field in */etc/passwd*—and replace it when you are finished with this procedure.)

For example, to change the password of the *bin* account to "2themoon" you enter:

`passwd bin`

You see:

```
New password:
```

Now enter the string "2themoon" and then press **`<Enter>`**. The string "2themoon" is not displayed as you type it. Next you see:

```
Re-enter password:
```

Enter the string "2themoon" again and then press **`<Enter>`**. The string is still not displayed as you type it.

Examine the entry for the *bin* account in the file */etc/passwd*. You should see something like this:

```
bin:SaXub4uaL5NP:2:2:System Tools Owner:/bin
```

The second field (between the first and second colons) is the encrypted version of the password "2themoon." (What you see may be different, even with the same password, depending on the "seed" the system uses to encrypt the password.)

5. Edit the file */etc/d_passwd*. In the file, place lines in the format:

   shell:password:

   shell is the command interpreter (shell) you wish to have a password, and *password* is the encrypted password. Make sure that all "shells" used in */etc/passwd* (the seventh and final field) are listed in this file, including those for UUCP, PPP, SLIP, and so on.

   For example, this command assigns the password "2themoon," which you encrypted in the previous step, to all C shell users who log in on the ttys specified in */etc/dialups*:

   `/bin/csh:SaXub4uaL5NP2:`

   You must place a colon at the end of the encrypted password, and you must enter the shell program pathname exactly as it appears in */etc/passwd*.

   Write the file and exit from the editor.

6. Make sure the files have appropriate permissions by issuing the command:

```
chmod 640 /etc/d_passwd /etc/dialups
```

7. Remove the password you assigned to the system account in step 4. To do this, edit the file */etc/passwd* and remove the string of characters in the second field. Return this field to the same state as when you began this procedure.

Now, whenever C shell users log in on the ttys specified in */etc/dialups*, they are prompted for the system password "2themoon" in addition to their account password.

Note that you must make similar entries for any other login shells used on your system such as */bin/ksh*, */usr/local/bin/bash*, and */usr/bin/tcsh*.

Creating a Shadow Password File

A "shadow" password file is simply a copy of the standard password file, but it is not accessible by non-privileged users. In the standard configuration, the */etc/passwd* file is publicly readable. Since the */etc/passwd* file contains the encrypted versions of the users' passwords, anyone can make a copy and attempt decryption of the passwords for malicious purposes. By using a shadow password file, you prevent intruders from attempting to decrypt your passwords.

The shadow password file is called */etc/shadow*. Once shadow passwords have been initialized, the password field in each */etc/passwd* entry is replaced by an "x" character.

To initialize */etc/shadow* (and thus invoke shadow passwords), run the pwconv(1M) command. Once this command has been run, shadow passwords are in effect. All standard password tools work transparently with shadow passwords. The difference should not be noticeable to your users, except that they cannot see the encrypted passwords in the */etc/passwd* file.

One difference in system operation is that older applications cannot get the proper value of *pw_passwd* from the getpwent(3C) and getpwnam(3C) library calls. This primarily affects "screen saver" programs, unless they have root privileges.

Note: Shadow passwords work differently with NIS. See the shadow(4) reference page for details on the use of shadow passwords with NIS.

Password Aging

The password aging mechanism forces users to change their passwords periodically. It also prevents a user from changing a new password before a specified time interval. You can also force a user to change his or her password immediately.

Realistically, password aging forces users to adopt at least two passwords for their accounts. This is because, when password aging is enforced, most users alternate between two passwords that they find easy to remember rather than inventing new passwords every time their old ones expire. IRIX does not provide a utility that determines whether users are choosing from a set of passwords and, if so, then forces them to choose completely different passwords.

Note: Password aging is not supported for NIS entries (see passwd(4)).

Password Aging With the *passwd* Command

To set the maximum number of days that can elapse before a user must change his or her password, use the passwd(1) command with the following syntax:

passwd –x *max name*

where *max* is the maximum number of days the password is valid for the user *name*. For example, this command forces user *alice* to change her password every two weeks (14 days):

```
passwd –x 14 alice
```

If you set *max* to 0, the user must change her password when she next logs in, but thereafter password aging is not in effect for her. If you set **-x** to -1, password aging is turned off immediately for that user.

You can also set the minimum time that must elapse before users are allowed to change their passwords. This is useful to prevent users from changing their passwords, then changing them back to their old passwords immediately. For example:

```
passwd –x 14 –n 7 ralph
```

This forces user *ralph* to change his password every fourteen days and prevents him from changing it more frequently than once every seven days. Note that if you set the minimum value greater than the maximum value, the user may not ever change his or her password.

To force users to change their passwords immediately, use the **-f** option. For example:

```
passwd -f trixie
```

Using Password Aging Manually

Another way to enforce password aging is to edit the */etc/passwd* file and insert the appropriate information after the password fields in the desired account entries.

Password aging information is appended to the encrypted password field in the */etc/passwd* file. The password aging information consists of a comma and up to four bytes (characters) in the format:

,Mmww

The meaning of these fields is as follows:

,	The comma separates the password and the aging information.
M	The *M*aximum duration of the password.
m	The *m*inimum time interval before the existing password can be changed by the user.
ww	The week (counted from the beginning of 1970) when the password was last changed and two characters, *ww*, are used. You do not enter this information. The system automatically adds these characters to the password aging information.

All times are specified in weeks (0 through 63) by a 64-character alphabet. The following chart shows the relationship between the numerical values and character codes. Any of the character codes can be used in the four fields of the password aging information. Table 4-1 lists the password aging codes and their meanings.

Table 4-1 Password Aging Character Codes

Character	Number of Weeks
. (period)	0 (zero)
/ (slash)	1
0 through 9	2 through 11

Table 4-1 (continued) Password Aging Character Codes

Character	Number of Weeks
A through Z	12 through 37
a through z	38 through 63

Two special cases apply for the character codes:

- If *M* and *m* are equal to zero, the user is forced to change the password at the next login. No further password aging is then applied to that login account.
- If *m* is greater than *M*, only *root* is able to change the password for that login account.

The following example shows the password aging information required to establish a new password every two weeks (0) and to deny changing the new password for one week (/) for user *ralph*:

```
ralph:RSOE2m.E,0/:100:1:Ralph P. Cramden:/usr/people/ralph:
```

After *ralph*'s first login following the change, the system automatically adds the two-character, "last-time-changed" information to the password field:

```
ralph:RSOE2m.E,0/W9:100:1:Ralph P. Cramden:/usr/people/ralph:
```

In this example, *ralph* changed his password in week W9. To force *ralph* to change his password at the next login (and to cause this only once), you can add the code **,..** to the password field:

```
ralph:RSOE2m.E,..:100:1:Ralph P. Cramden:/usr/people/ralph:
```

After *ralph* changes his password, the system automatically removes the aging code (**,..**) from the password field. To prevent *ralph* from changing his password, use the code `,./`. Edit the */etc/passwd* file and add a comma, period, and slash to the password field:

```
ralph:RSOE2m.E,./:100:1:Ralph P. Cramden:/usr/people/ralph:
```

Now only *root* can change the password for the *ralph* account. If *ralph* tries to change the password, he sees the message `permission denied`.

Using pwck to Check the Password File

From time to time, you should run the pwck(1M) utility to scan the password file. This program reads the file and checks each entry for completeness and notes any inconsistencies. The password checks include validation of:

- the number of fields in each entry
- the login name
- the user ID number
- the group ID number
- the login directory
- the executed program

The default password file to be checked is */etc/passwd*. If shadow passwords (described in "Creating a Shadow Password File" on page 89) are enabled, the */etc/shadow* file is checked.

Similarly, the grpck(1M) command verifies all entries in the */etc/group* file. The default group file to be checked is */etc/group*. With either command, an alternate file may be specified on the command line.

Login and Account Administration

This section describes how to control special and login accounts. Special accounts are used by the system to perform specific system functions, and login accounts are user accounts allowing general-purpose system access.

Special Accounts

Special accounts are used by daemons to perform system functions, such as spooling UUCP jobs and print requests. Because key files are owned by these accounts, someone who has obtained access to one of the accounts, or was able to start a daemon on your system, could partially breach security. Partially, because ownership of the various system files is distributed among the special accounts.

Guard access to all the special accounts as you would the *root* account. Either assign passwords to these accounts, or lock them using one of the methods described in "Locking Unused Logins" on page 94.

Following is a list of all the administrative and special accounts on the system and what they are used for:

root	This login has no restrictions, and it overrides all other logins, protections, and permissions. It allows you access to the entire operating system. The password for the *root* login should be very carefully protected.
sys	This login has the power of a normal user login over the files it owns, which are in */usr/src*. Its login should be disabled.
bin	This login has the power of a normal user login over the files it owns, which are throughout the system. Its login should be disabled.
adm	This login has the power of a normal user login over the files it owns, which are located in */var/adm*. You may *su* to the adm login. This login should be disabled.
uucp	This login owns the object and spooled data files in */usr/lib/uucp* and */etc/uucp*.
nuucp	This login is used by remote workstations to log into the system and initiate file transfers through */usr/lib/uucp/uucico*.
daemon	This login is the system daemon, which controls background processing. Its login should be disabled.
lp	This login owns the object and spooled data files in */var/spool/lp*. Its login should be disabled unless the system is a print server.

Locking Unused Logins

If a login is not used or needed, disable (lock) the login. You should not remove the account, though, because of the danger of reusing the UID in the future. User ID numbers are meant to be permanently associated with the person who used the account. If you reuse the UID number, the new user may find files that belonged to the previous owner of the ID number. These files may contain "trojan horse" programs that could damage your system. You may remove the user's home directory and files (after making a backup), but you should never remove an entry from your */etc/passwd* file.

There are two ways to lock an account. The first is using the *passwd* command with the **-l** option. For example, the current entry in */etc/passwd* for the user *jones* might look like this:

```
jones:6.D/N3ZFGmq7U:3333:10:Jeremiah Jones:/usr/people/jones:/bin/tcsh
```

Enter the following command:

`passwd -l jones`

and the entry becomes:

```
jones:*LK*:3333:10:Jeremiah Jones:/usr/people/jones:/bin/tcsh
```

This command changes the password field of the entry in */etc/passwd* for account *jones* to `*LK*`. This blocks all logins to that account.

The second way to lock an account is by editing the password file directly. Change the password field to any string of characters that is not used by the password encryption program to create encrypted passwords. The *passwd* command with the **-l** option uses the string `*LK*`. You can use other strings to lock accounts.

For example, you can use a descriptive phrase such as "LOCKED;" to remind you that the account was deliberately disabled:

```
ralph:LOCKED;:100:1:Ralph P. Cramden:/usr/people/ralph:
```

The semicolon is not used in an encrypted password and causes the account to be locked. The text "LOCKED" is merely to remind you that the account is locked.

Another common method of disabling a password is to put an asterisk (*) in the password field. The default IRIX */etc/passwd* file disables some unused logins in this manner. Be sure to check your */etc/passwd* file to be sure all logins have passwords or are disabled.

System Login Options

You can set the following login options to enhance security:

- Restrict *root* logins to a specific device, typically the system console.
- Specify the number of times an attempt to log in can fail before the login process is disabled.
- If the login process is disabled, specify how long before it can be resumed.

- Specify whether to maintain a log of logins and what information to store: all logins or only those that were unsuccessful.
- Specify whether to force a user who does not have a password to choose one immediately upon logging in.
- Specify whether or not to display to the user on login the date and time that user last logged in.

Login options are set in the file */etc/default/login,* which is a normal text file. The file contains one option specification per line. The options are described in the rest of this section.

Because the login procedure is your system's main defense against unauthorized access, login options are important. For example, you can determine whether someone is trying to break into your system from a pattern of failed login attempts recorded in */var/adm/SYSLOG* (when logging is enabled).

The best way to keep a system secure is to slow down attempts to guess passwords and account names. The login options described in this section add delays to unsuccessful login attempts, which drastically slows down the process of randomly guessing passwords.

See the login(1) reference page for further details.

Note that the visual login process clogin(1) does not provide these security options. To use the login security functions, you must turn off *clogin* and use the standard login processes, getty(1) and login(1). Use *chkconfig* to turn off the *visuallogin* and *xdm* configuration variables. See "IRIX Admin: System Configuration and Operation" and the visuallogin(4) reference page for information about turning the visual login process on and off. You may also use *chkconfig* to set the *noiconlogin* variable to disallow logging in using the user icons in *clogin*.

Restricting root Logins

You can restrict *root* logins to a single device, forcing *root* users to either use that device or use the *su* command (thereby leaving a trail in */var/adm/sulog*). For example, edit */etc/default/login* to include the following line to restrict root logins to the system console:

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

Note: Do not name */dev/syscon* or */dev/systty* as the device!

Maximum Login Attempts (MAXTRYS)

MAXTRYS is the number of times you let a login attempt fail before suspending the login. Setting this parameter slows attempts by unauthorized persons to break into a system. A common method of breaking into a system is to try to guess the password of a known account. This method is most successful if the person trying to break in knows the names of as many accounts as possible, and can make guesses very quickly. If you introduce a delay in the login process after a certain number of failed login attempts on the same *tty* line, you can make it much more time-consuming to guess a correct password.

To set the maximum number of login attempts, edit the file */etc/default/login*. Place a line similar to this in the file:

```
MAXTRYS=3
```

This sets the maximum number of login attempts to three. The system default, without this option set, is five.

When the maximum number of login attempts is exceeded, the *login* program sleeps for a certain number of seconds (the DISABLETIME variable described in the next section), thus preventing further login attempts on that line for a while. The system default delay (DISABLETIME) is 20 seconds.

Following is an example login attempt that is disabled after three retries:

```
login: guest
password:
Login incorrect
login: guest
password:
Login incorrect
login: guest
password:
Login incorrect
```

At this point, no further login prompts are displayed until the period of time specified by DISABLETIME has passed.

Length of Time to Disable a Line (DISABLETIME)

Use this option along with the *MAXTRYS* option. To set the number of seconds after a certain number of unsuccessful login attempts that a line is disabled, edit the file */etc/default/login* and add a line similar to this:

```
DISABLETIME=30
```

This disables a line for 30 seconds. You can choose any value you consider appropriate for your system. The system default is 20 seconds.

Recording Login Attempts

You can record both successful and unsuccessful login attempts in the file */var/adm/SYSLOG*. To record all attempts to log in, place this line in the file */etc/default/login*:

```
SYSLOG=ALL
```

To record only unsuccessful attempts, place this line in the *login* file:

```
SYSLOG=FAIL
```

A large number of failed logins, especially with the same account name, may indicate that someone is trying to break into that account and thus into the system.

Forcing a Password

To force users who do not have passwords for their accounts to choose their passwords immediately, add this line to the file */etc/default/login*:

```
PASSREQ
```

Or, insert the following entry instead:

```
MANDPASS=YES
```

to prevent users from logging in if they do not already have a password.

Displaying the Last Login Time

Users can help maintain system security by noticing unauthorized use of their accounts. By default, the most recent login date, time, and the name of the terminal line (*tty* name) or remote host from which the user logged in is displayed on login. This login attempt information is recorded in files, one per user account and with the same name as the account, in the directory */var/adm/lastlog*.

Users can stop the last login information from being displayed by having a *.hushlogin* file in their home directory, but they should be discouraged from doing so. Remind them periodically to look at the information each time they log in for any unusual information.

Set-UID and Set-GID Permissions

The set user identification (set-UID) and set group identification (set-GID) permissions must be used very carefully. When a user runs an executable file that has either of these permissions, the system gives the user the permissions of the owner of the executable file. You can add these permissions to any executable file with the chmod(1) command.

Set-UID and set-GID programs have legitimate uses, but because they are potentially harmful, there should be very few of them on your system. Beware of programs in publicly writable directories (such as */tmp*, */usr/tmp.O*, */var/tmp*, and */usr/spool/uucppublic*) that have the same name as common systems files (such as *vi* and *rm*). One reason the PATH environment variable of the *root* account does not include the current directory (as does the default PATH of most other users) is so that *root* won't accidentally execute such "booby-trap" programs.

System security can be compromised if a user copies another program onto a file with `-rwsrwxrwx` permissions. To take an extreme example, if the *su* command has the write access permission allowed for others, anyone can copy the shell onto it and get a password-free version of *su*.

The following sections provide some example commands that identify files on the system with set-UID permissions. For more information about the set-UID and set-GID bits, see the chmod(1) and chmod(2) reference pages.

Checking for Set-UID Files Owned by root

The following command line lists all set-UID files owned specifically by *root*:

```
find / -user root -perm -4000 -print
```

The results of this command are printed on the screen. All paths are checked starting at /, including all mounted directories. A great number of files will be found. It is up to you to scan these files for any unusual names. One possibility is to direct the output of this program to a file soon after installation and compare the results with later outputs. If this command reports any unusual files, investigate them immediately.

A suspicious file might turn up like this:

```
-r-sr-xr-x 1 root  bin   38836 Aug 10 16:16 /usr/bin/at
-r-sr-xr-x 1 root  bin   19812 Aug 10 16:16 /usr/bin/crontab
-r-sr-xr-x 1 root  bin   27748 Aug 10 16:16 /usr/bin/shl
---s--x--x 1 root  sys   46040 Aug 10 15:18 /usr/bin/ct
-r-sr-sr-x 1 root  bin   33208 Aug 10 15:55 /usr/lib/lpadmin
-r-sr-sr-x 1 root  bin   38696 Aug 10 15:55 /usr/lib/lpsched
---s--x--- 1 root  user  45376 Aug 18 15:11 /usr/jbond/bin/sh
-r-sr-xr-x 1 root  sys   11416 Aug 11 01:26 /bin/mkdir
-r-sr-xr-x 1 root  sys   11804 Aug 11 01:26 /bin/rmdir
-r-sr-xr-x 1 root  bin   12524 Aug 11 01:27 /bin/df
-rwsr-xr-x 1 root  sys   21780 Aug 11 01:27 /bin/newgrp
-r-sr-sr-x 1 root  sys   23000 Aug 11 01:27 /bin/passwd
-r-sr-xr-x 1 root  sys   23824 Aug 11 01:27 /bin/su
```

In this example, the user *jbond* has a personal copy of */bin/sh* and has made it set-UID to *root*. This means that anyone in the group *user* can execute */usr/jbond/bin/sh* and become the superuser.

Checking for Set-UIDs in the root Filesystem

The following command line reports all files with a set-UID for the *root* filesystem (not just those owned by *root*) on EFS filesystems:

```
ncheck -s /dev/root | xargs ls -ld | cut -f2 | grep -v ~/dev/
ls -l `/etc/ncheck -s /dev/root | cut -f2 | grep -v dev`
```

The ncheck(1M) command, by itself, can be used on a mounted or unmounted file system. Only the superuser may use *ncheck*. The normal output of the *ncheck -s* command includes special files. Here, the *grep* command removes device files from the output. This filtering is applicable only for the *root* filesystem. The output of the modified *ncheck* is then used as an argument to the *ls* command. The filesystem must be mounted for the *ls* command to succeed. In this example output, nothing looks suspicious:

```
-r-sr-xr-x  1 root   bin    12524   Aug 11 01:27 /bin/df
-rwxr-sr-x  1 root   sys    32272   Aug 10 15:53 /bin/ipcs
```

```
-r-xr-sr-x  2 bin    mail   32852   Aug 11 01:28 /bin/mail
-r-sr-xr-x  1 root   sys    11416   Aug 11 01:26 /bin/mkdir
-rwsr-xr-x  1 root   sys    21780   Aug 11 01:27 /bin/newgrp
-r-sr-sr-x  1 root   sys    23000   Aug 11 01:27 /bin/passwd
-r-xr-sr-x  1 bin    sys    27964   Aug 11 01:28 /bin/ps
-r-xr-sr-x  2 bin    mail   32852   Aug 11 01:28 /bin/rmail
-r-sr-xr-x  1 root   sys    11804   Aug 11 01:26 /bin/rmdir
-r-sr-xr-x  1 root   sys    23824   Aug 11 01:27 /bin/su
-r-xr-sr-x  1 bin    sys    21212   Aug 10 16:08 /etc/whodo
```

For XFS filesystems, use the *find* command:

`find / -perm -4000 -print`

Checking Set-UIDs in Filesystems Other Than root

This example uses the *ncheck* command to examine the *usr* filesystem (*/dev/usr*, assuming a single-disk system with default partitioning) for files that have set-UID permissions:

`/etc/ncheck -s /dev/usr | cut -f2`

In this partial example below, complete pathnames for the files start with */usr*. */usr* is not part of the *ncheck* output.

In this sample output, the program */usr/people/jbond/bin/sh* should be investigated. This program is the only one that is not found in a system directory. It is a command shell residing in a user's home directory. Users should, in general, not possess set-UID binaries.

```
/dev/usr:
/bin/at               /bin/uux
/bin/crontab          /lib/mv_dir
/bin/shl              /lib/expreserve
/bin/sadp             /lib/exrecover
/bin/timex            /lib/accept
/bin/cancel           /lib/lpadmin
/bin/disable          /lib/lpmove
/bin/enable           /lib/lpsched
/lib/reject           /lib/lpshut
/lib/sa/sadc          /bin/lp
/lib/uucp/uucico      /bin/lpstat
/lib/uucp/uusched     /bin/ct
/bin/uucp             /bin/cu
/bin/uuname           /lib/uucp/uuxqt
```

```
/bin/uustat           /usr/people/jbond/bin/sh
```

General File and Directory Permissions

Be conservative when establishing or changing permission bit settings on all files and directories. The safest settings do not allow write access, but where this is not possible, it may be possible to limit write access to the owner of the file or directory, or at least just to the owner and the group.

You should not be running the *rfindd* daemon, because it allows external access to your file, directory, and permissions listing. Use *chkconfig* to turn this daemon off if it is on. Refer to rfindd(1M) and chkconfig(1M) for more information.

Refer to the chmod(1) reference page for a discussion on setting the sticky bit on such directories as */tmp* to restrict removal and renaming of files.

The following files and directories are universally available for read and write access on IRIX as shipped. Depending on your site requirements, you may wish to change the permissions on these files to be more restrictive.

Caution: Restricting permissions on historically open directories, such as */tmp*, */usr/tmp.O*, and */var/tmp* (linked to */usr/tmp*), can cause serious malfunctions in many programs, applications, and system utilities that write temporary files on behalf of users in these directories. Below is a partial list of such directories.

- */tmp*
- */usr/demos/.xsession*
- */usr/Insight/tmp*
- */usr/Insight/tmp/ebtpriv*
- */usr/Insight/tmp/ebtpub*
- */usr/Insight/tmp/install.insight.log*
- */usr/lib/emacs/maclib*
- */usr/lib/showcase/fonts*
- */usr/lib/showcase/images*
- */usr/lib/showcase/models*

- */usr/lib/showcase/templates*
- */usr/tmp.O*
- */var/spool/locks*
- */var/spool/uucppublic*
- */var/tmp*

Accounts Shipped Without Passwords

The following accounts in your default */etc/passwd* file are shipped without passwords. You should create passwords for these accounts immediately.

- demos
- guest
- lp
- nuucp
- root
- tour
- tutor
- 4Dgifts

Caution: Creating passwords on historically open accounts, such as *lp*, may cause certain related applications or operations to fail.

Security File and Command Reference

This section summarizes in two tables some IRIX files and commands that establish and control security. Table 4-2 lists the IRIX files concerned with security and Table 4-3 lists security-related commands.

Table 4-2 IRIX Security Files

File	Purpose	Reference
/etc/default/login	Control login actions	login(1)
/etc/default/su	Define *su* command defaults	su(1M)
/etc/passwd	Store password and account information	passwd(1), passwd(4)
/etc/shadow	Hide password information	shadow(4), pwconv(1M)
/var/adm/sulog	Log *su* command usage	su(1M)
/var/adm/SYSLOG	Log system messages	syslogd(1M)

Table 4-3 IRIX Security Commands

Command Example	Purpose	Reference
arp -a	Display current ARP entries	arp(1M), arp(7P)
crypt *password*	Encode/decode input/output	crypt(1)
last	Indicate last logins of users and terminals	last(1)
ncheck(1M)	Generate pathnames from i-numbers	ncheck(1M)
passwd	Change password	passwd(1), passwd(4)
ps -elf	Display a full, long list of every process currently running	ps(1)

Table 4-3 (continued) IRIX Security Commands

Command Example	Purpose	Reference
pwck	Report inconsistencies in */etc/passwd* file	pwck(1M), passwd(4)
sar	System activity reporter	sar(1), sar(1M)
satd	Reliably save the system audit trail	satd(1M) and "Placing the Audit Files" on page 143
vi -x	Edit encrypted file	vi(1), crypt(1)
w	Display users logged in with current activity	w(1)
who	Display users logged in, their tty, and time of login	who(1)