Basic System Security [2]

Maintaining security on UNICOS systems is largely a matter of vigilance on the part of the system administrator, who should maintain constant surveillance for potential security problems and for evidence of past security breaches. The UNICOS operating system includes programs that provide the necessary tools for the creation of a set of procedures that lets you automate much of the daily work of monitoring system security. This chapter discusses security issues in four areas: system security (ensuring that the super-user privileges are safe), user security, partition security, and tape device access.

2.1 Related basic system security documentation

The following documentation contains more detailed information about the material presented in this chapter:

- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022: `diskusg`(8) man page
- *General UNICOS System Administration*, Cray Research publication SG–2301
- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011: `chown`(1), `du`(1), `find`(1), `login`(1) `su`(1), and `umask`(1) man pages

2.2 Super-user privileges

As in standard UNIX systems, in the UNICOS operating system, the user identification number (user ID) of 0, associated with the account named `root`, has special privileges and may override the security features that govern the activity of normal users. Such a user is referred to as a *super user*, and the super user's powers allow the administrator great flexibility in responding to system problems and keeping the system running smoothly. The dominant security concern for a UNICOS system administrator is ensuring that access to super-user privileges remains solely in the hands of the administrator and the administrator's staff. Failure to guard this access allows unauthorized users to acquire super-user privileges. At best, one user could then look at other users' sensitive files without authorization and, at worst, an outside intruder (knowingly or unknowingly) could cause damage to the entire system.

2.2.1 Password security for super-user

The password to the super-user (`root`) account is the first line of defense against security breaches. Anyone logging in as `root` or using the `su` command to acquire super-user privileges uses this password.

To maintain secure access to the root account, you should use the following steps:

- Ensure that the `root` password is not obvious and is very difficult to guess. Do not use a normal word in any language that might be known to a majority of the system's users. Additionally, capitalizing a random letter or two (not the first letter of the password), or including a punctuation character or a numeral in the password, or both, helps to keep super-user privileges safe from an intruder who is trying to guess the `root` password.
- Change the root password frequently, at least once a month.
- Do not write down the `root` password.
- Ensure that the `root` password is known to as few people as possible; generally, these should be the system administrator and the administrator's staff.

You can monitor the use of the `root` password, and catch potential security breaches, by checking the `/usr/adm/sulog` file. (For information about system logs, see Chapter 9, page 225.) You can compare the log entries against the names of users known to have valid authorization, alerting the administrator to unauthorized super users (a security breach) or users who are repeatedly trying to gain super-user privileges (a security risk).

2.2.2 Physical security

A person who has access to the system workstation (SWS) and a knowledge of how to halt and reboot the system could do so, and thus, acquire unauthorized super-user privileges.

To guard against this possibility, your SWS and your system itself should be physically accessible only to those persons who genuinely need that access, and your SWS should not be left unattended while you are logged into the system. If this is not possible, your SWS should at least be monitored to prevent unauthorized persons from trying to enter commands on the system console.

Store all removable media in a secure location. Always store backup tapes and cartridges and other media in a location different than your system. You also should make certain that any external media is in a physically secure location.

2.2.3 setuid programs

An executable UNICOS program may have the `setuid` (set user ID) bit in its permissions code set, indicating that whenever any user executes the program, the program runs with an effective user ID of the owner of the file. Thus, any program that `root` (user ID 0) owns and has the `setuid` bit on can override normal permissions, regardless of who executes the program.

This feature is useful and necessary for many UNICOS utilities and commands, but it can be a potential security problem if an astute user discovers a way to create a copy of the shell owned by `root`, with the `setuid` bit on. To avoid this possible security breach, you should make regular checks of all disk partitions on the system for programs that have a `setuid` or a `setgid` (which is the set group ID) of 0.

The `find`(1) command can generate a list of all `setuid` or `setgid` 0 files on the system (if all file systems are mounted), as follows:

```
# find / -user 0 -perm -4000 -o -group 0 -perm -2000 -print | xargs sum
```

Compare this list against a list of known `setuid` or `setgid` 0 programs. Any new `setuid` or `setgid` 0 programs that are not on the known list and whose creation you cannot account for may indicate a security breach.

As administrator, you should check the list of known `setuid` or `setgid` 0 programs regularly to ensure that none have been modified since the last check and that any modifications that have been made are known (that is, were made by you or a member of your staff). Unknown modification of a `setuid` or `setgid` 0 program may indicate a security breach. To generate a checksum and block count list of a file, you can use the `sum`(1) command; to do this, you can pipe the preceding `find` command line through the `sum` command as follows and then compare any changes in sizes:

`| sum >` *filename*

To ensure that write permission on each file is properly restricted, you also should check the list of known `setuid` or `setgid` 0 programs.

Because checking the entire system for `setuid` or `setgid` 0 programs uses a lot of I/O and CPU time, you should perform this check during off-peak hours.

To make the task less obtrusive, use the `cron`(1) or `at`(1) command to perform the check automatically.

2.2.4 `root` PATH

The `PATH` environment variable consists of a list of the directories that the shell searches for typed commands. This means that the `PATH` for the `root` account must have the following security features:

- It must never contain the current directory (.).
- All directories listed in the `root` `PATH` must never be writable by anyone other than `root`.

The `root` `PATH` is set in two separate places:

- The `.profile` file sets the `PATH` for `root` whenever `root` logs in on the system console.
- The `su`(1) command changes the `PATH` after a user has entered the `root` password to successfully assume super-user privileges.

To make sure that the path has not been changed in either place since the last approved change, you should monitor both places occasionally.

Keeping the current directory out of the `root` `PATH` is somewhat inconvenient; super users must remember to precede the names of any programs or scripts they want to run from their current directory with `./`, as in `./newprogram`, because the shell does not search the current directory for a command name. However, convenience should not take precedence over system security. Failure to follow these guidelines leaves the system open to a security breach.

For example, suppose a knowledgeable user creates a program that mimics a commonly used system utility, such as `ls`. In addition to performing the expected system function (listing the files in the current directory), the new `ls` utility makes a copy of a program such as `sh` and turns on the `setuid` bit on the copy. An unsuspecting super user who has the current directory in `PATH`, having changed directories to a user's directory and inadvertently run the bogus `ls`, then creates a `setuid` 0 shell, which gives anyone executing it complete control over the system.

2.3 User security

In addition to general system security, you should ensure that the files system users own are secure from examination and modification by other users.

2.3.1 `umask` command

The system default umask value is usually set in `/etc/profile` by using the `umask`(1) command. It lets you choose the permissions that typically will be set when users create new files (for example, a umask value of `027` means that the group and other write permissions and the other read and execute permissions are not set when a user creates a file). For possible umask values and descriptions, see the `umask`(1) man page.

Generally, only the owner of the file should have write permission, which makes a default umask value of `022` appropriate. If members of a given user group should not be able to read the files of other user groups, you should use a umask value of `026` to remove other read permission.

You should choose a umask value that restricts default access permissions to a level appropriate to the desired security of the system. However, because users can override the default value by using the `umask` command themselves, do not make the default umask value too stringent, because users may find that the default value interferes with their work. For instance, if two users are working on a joint project, and each needs access to the other's files, they may want to change their umask value to open their files. As an alternative, they may want to use the groups mechanism; see Section 2.3.3, page 16.

2.3.2 Default `PATH` variable

The default `PATH` variable for the system's users is set in the `/etc/profile` and `/etc/cshrc` files. It specifies the system directories that will be searched for command names typed by the users.

The users expect to be able to execute programs in the current directory without preceding the program name with `./`, which explicitly indicates the current directory. However, many UNICOS systems traditionally place the current directory first in the `PATH`, which can make the users vulnerable to a security breach. The current directory should thus be the last entry in the default `PATH`, after the normal system directories.

2.3.3 User groups

You can enhance user security by the careful placement of users into groups. Generally, when deciding on the placement of users into groups, you should use factors external to the system. Some examples might be the following:

- Members of a specific software project
- Accounts for a client company purchasing system time
- Intercompany divisions

Having many groups, each containing a small number of users, is safer than having fewer groups, each with large numbers of users who have access to each other's files. Members of most logical groups (for example, members of a software development project) want to share files with one another, and the default umask should permit this.

To prevent inappropriate sharing of data, you should create a group that has only one user in it, rather than create a default "other" or "miscellaneous" group for users who do not fit elsewhere. Because users may belong to more than one group, and groups are active simultaneously, you also may choose to create a separate group for each individual user at the time you create the account, and then add users to additional logical groups as necessary.

2.3.4 File-owner fraud

Neither the listed owner ID of a file nor its location in the directory tree always leads to the actual creator and owner of the file. That is, users tend to think of the files residing in their home directory as their only files, even though they may own files in another home directory, such as those being used for a project that involves several other users. Files that reside in one user's home directory tree may also be owned by another user.

Users may become confused by this situation and then use the `chown`(1) command to change the ownership of some of their files to another user (most likely one who will cooperate and give the file back when requested). To get a general idea of the users who trade ownership of files, you can use the `diskusg`(8) and `du`(1) commands together.

2.3.5 Login attempts

Unauthorized users might try to gain access to the system by making repeated attempts to log in. To help prevent such attempts, you can configure the

number of bad login attempts that will be allowed before the `login` terminates. By default, the system will allow an unlimited number of bad login attempts. To put a limit on such attempts, edit the `/etc/config/confval` file (see `login`(1)).

2.4 Partition security

When administered properly, the UNICOS file system should provide adequate protection for user and system files. To enhance system security, however, mount file systems only when they are needed. In particular, if there are users who will be allowed dedicated time on your system, you can provide extra protection for those accounts by not mounting their files during nondedicated time or by not mounting the file systems that contain other users' accounts during dedicated time. (For more information about file systems, see Chapter 5, page 51.)

To prevent users from accessing disk partitions directly, without going through the UNICOS file system, the disk device nodes in `/dev/dsk` and `/dev/rdsk` must never be readable or writable by anyone other than `root`.

Example:

```
brw-------   1 root     root       0,245 Nov 28 20:24 bk_udb
brw-------   1 root     root       0,247 Nov 28 20:24 bkroot
brw-------   1 root     root       0,246 Nov 28 20:24 bkusr
```

2.5 Tape device access

For CRAY J90 systems, you should be using the tape daemon character-special tape interface. The character-special tape interface provides unstructured access to the tape hardware similar to the traditional UNIX method of accessing tape devices. It is useful in performing specific tasks, such as the following:

- System administrators use the interface for routine tape manipulations such as copying. To manage their tapes, they can use standard UNIX commands and `ioctl`(2) requests.
- Programmers use the interface to develop file management applications.

For more information on tape devices, see the *Tape Subsystem Administration*, Cray Research publication SG–2307, and the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.