Basic System Security [3]

This section discusses security issues in the following areas: system security (ensuring that the super-user privileges are safe), user security, partition security, and tape device access.

3.1 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

3.2 Monitoring system security

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. Fortunately, UNICOS includes programs that provide the necessary tools to create a set of procedures that allows you to automate much of the daily work of monitoring system security. This section discusses security issues in three areas: system security (ensuring that the super-user privileges are safe), user security, and partition security.

Note: This section does **not** apply to systems using the UNICOS multilevel security (MLS) feature systems. The MLS feature provides mechanisms to protect both system integrity and sensitive information. Design specifications for the UNICOS MLS feature were derived from the U.S. Department of Defense (DoD) evaluation criteria for trusted computer systems.

3.2.1 Super-user privileges

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 governing 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 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 an unauthorized user 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.

3.2.1.1 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`(1) utility to acquire super-user privileges uses this password.

Cray Research recommends the following steps to maintain secure access to the `root` account:

- The `root` password should not be obvious and should be 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.
- The `root` password should be changed frequently, at least once a month.
- The `root` password should never be written down anywhere.
- The `root` password should be known to as few people as possible. Generally, these should be the system administrator and the administrator's staff.

Use of the `root` password can be monitored, and potential security breaches caught, by compiling the `su` utility so that it logs each use of the utility in the `/usr/adm/sulog` file. The administrator can then use the `grep`(1) utility to generate periodic lists of successful and unsuccessful attempts to assume super-user privileges by use of `su`. These lists can be compared 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).

3.2.1.2 Physical security

A person with access to the SWS and ION consoles 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, Cray Research recommends that the SWS and the ION consoles and the system itself be physically accessible only to those persons with genuine need for that access. If this is not possible, they should at least be monitored to prevent unauthorized persons from attempting to enter commands on the system console.

3.2.1.3 setuid programs

An executable UNICOS program may have the setuid 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 is owned by `root` (user ID `0`) and has its setuid bit set is able to 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, the administrator should make regular checks of all disk partitions on the system for programs that have a setuid (or setgid) of 0.

The `find`(1) utility can generate a list of all setuid/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
```

This list may be compared against a list of known setuid/setgid 0 programs. Any new setuid/setgid 0 programs that are not on the known list and whose creation you cannot account for may indicate a security breach.

The administrator should check the list of known setuid/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 (in other words, were made by the system administrator or a member of the administrator’s staff). Unknown modification of a setuid/setgid 0 program may indicate a security breach.

Finally, the list of known setuid/setgid 0 programs should be checked to ensure that write permission on each file is properly restricted.

Because checking the entire system for setuid/setgid programs uses a large amount of CPU time, Cray Research recommends that this check be performed during off-peak hours. Use of the `cron`(8) or `at`(1) utility to perform the check automatically and to notify the administrator of any suspicious results should make the task unobtrusive.

3.2.1.4 `root PATH`

The `PATH` environment variable consists of a list of the directories searched by the shell 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) utility changes the `PATH` after a user has successfully entered the `root` password to assume super-user privileges.

Both places should be monitored from time to time to make sure they have not been changed since the last approved change known to the administrator.

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`(1). In addition to performing the expected system function (listing the files in the current directory), the new `ls`(1) utility makes a copy of a program such as `ksh`(1) and turns on the setuid bit on the copy. An unsuspecting super user with 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.

3.2.2 User security

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

3.2.2.1 The `umask` utility

The system default umask value is normally set in the `/etc/profile` file by using the `umask`(1) utility. It allows you to choose the permissions that will typically 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.

In general, only the owner of the file should have write permission, which makes a default umask value of 022 appropriate. If you do not want members of a given user group to be able to read the files of other user groups, using a umask value of 026 to remove other read permission is recommended.

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` utility themselves, do not make the default umask value too stringent, as 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 values so that, on any new files they create, the permissions will be more open.

3.2.2.2 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 having to precede the program name with `./` to explicitly indicate 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, as described in Section 3.2.2.4, page 22. The current directory should thus be the last entry in the default `PATH`, after the normal system directories.

3.2.2.3 User groups

User security can be enhanced by the careful placement of users into groups. In general, it is a good idea to use factors external to the system when deciding upon the placement of users into groups. 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 with access to each other's files. Members of most logical groups (for example, members of a software development project) may want to share files with one another, and the default umask should permit this.

To prevent inappropriate sharing of data, rather than create a default "other" or "miscellaneous" group for a user who does not fit elsewhere, you should create a group with only one user in it. Because users may belong to more than one group, and groups are active simultaneously, you may also 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.

3.2.2.4 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 involving several other users. Conversely, it may not be completely appropriate to count files that reside in one user's home directory tree but are owned by another user.

Users may realize this confusion and try to avoid a disk usage monitoring system by using the `chown`(1) utility 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). Nevertheless, `diskusg`(8) and `du`(1), when used together, provide a general idea of the users who are perennial problems.

3.2.2.5 Login attempts

Unauthorized users might attempt to gain access to the system by making repeated attempts to login. 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)).

3.2.3 Partition security

When administered properly, the UNICOS file system should provide adequate protection for user and system files. You can enhance system security, however, by mounting partitions 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 the file systems that contain other users' accounts.

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`.

3.3 Tape device access

For CRAY J90se systems, you should use 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.