UNICOS Multilevel Security (MLS) Feature [8]

This chapter describes the UNICOS multilevel security (MLS) feature for system and security administrators. The UNICOS security environment is established at UNICOS build time.

The information in this chapter is intended for system and security administrators. It is assumed that you have read and understood the information presented in the *UNICOS Multilevel Security (MLS) Feature User's Guide*, Cray Research publication SG–2111.

In general, a secure system must protect information from unauthorized disclosure and modification. A secure system must also be designed to guarantee that established security mechanisms are correctly implemented and consistently maintained. The UNICOS MLS feature provides mechanisms to protect both system integrity and sensitive information.

Warning: In previous releases of UNICOS operating system documentation, the term *Trusted UNICOS* was used to refer to the configuration that most closely approximated the B1 evaluated configuration of UNICOS release 8.0.2. In the UNICOS 10.0 release, this configuration is referred to as the *Cray ML-Safe configuration* of the UNICOS operating system. Although the Cray ML-Safe configuration of the UNICOS operating system is not an evaluated product, this configuration fully supports all functionality described in the B1 evaluation criteria.

Design specifications for the MLS feature were derived from the *Department of Defense Trusted Computer System Evaluation Criteria* (TCSEC). The UNICOS MLS feature implementation strategy is defined by the following basic DoD control objectives:

- Security policy
- Accountability
- Assurance

These Department of Defense (DoD) objectives are implemented by UNICOS MLS feature mechanisms that can be divided into the following security policies:

- Privilege and system management (also referred to as *trusted facility management*)

- Mandatory access control (MAC)
- Discretionary access control (DAC)
- Identification and authentication (I&A)
- Object reuse
- Installation and configuration
- Auditing

The following sections describe the mechanisms and associated procedures for the UNICOS MLS feature.

8.1 Overview of UNICOS security mechanisms

UNICOS security mechanisms are either part of the traditional UNICOS operating system (for example, file permission bits) or are enhancements to traditional UNICOS structures or functions (for example, mandatory access controls). These mechanisms and how they correlate to the *Trusted Computer System Evaluation Criteria (TCSEC)* are shown in Figure 5.

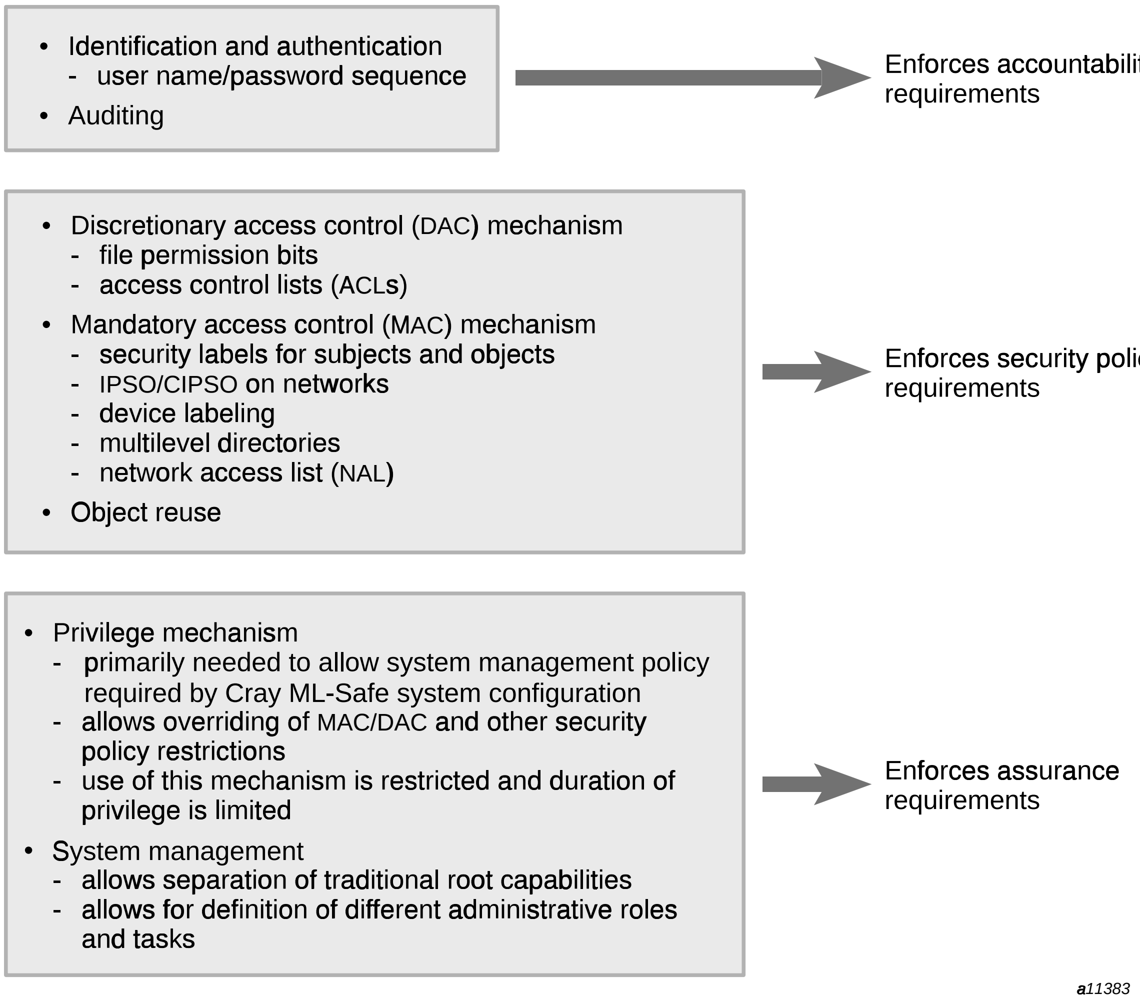
• Identification and authentication
- user name/password sequence
• Auditing
Enforces accountability requirements
• Discretionary access control (DAC) mechanism
- file permission bits
- access control lists (ACLs)
• Mandatory access control (MAC) mechanism
- security labels for subjects and objects
- IPSO/CIPSO on networks
- device labeling
- multilevel directories
- network access list (NAL)
• Object reuse
Enforces security policy requirements
• Privilege mechanism
- primarily needed to allow system management policy required by Cray ML-Safe system configuration
- allows overriding of MAC/DAC and other security policy restrictions
- use of this mechanism is restricted and duration of privilege is limited
• System management
- allows separation of traditional root capabilities
- allows for definition of different administrative roles and tasks
Enforces assurance requirements
a11383

Figure 5. UNICOS security mechanisms

Figure 6 presents a high-level overview of how the following security mechanisms interact in the sequence of UNICOS tasks:

- Identification and authentication (I&A)
- The security label portion of mandatory access controls (MAC) and discretionary access controls (DAC)
- The PAL-based privilege mechanism

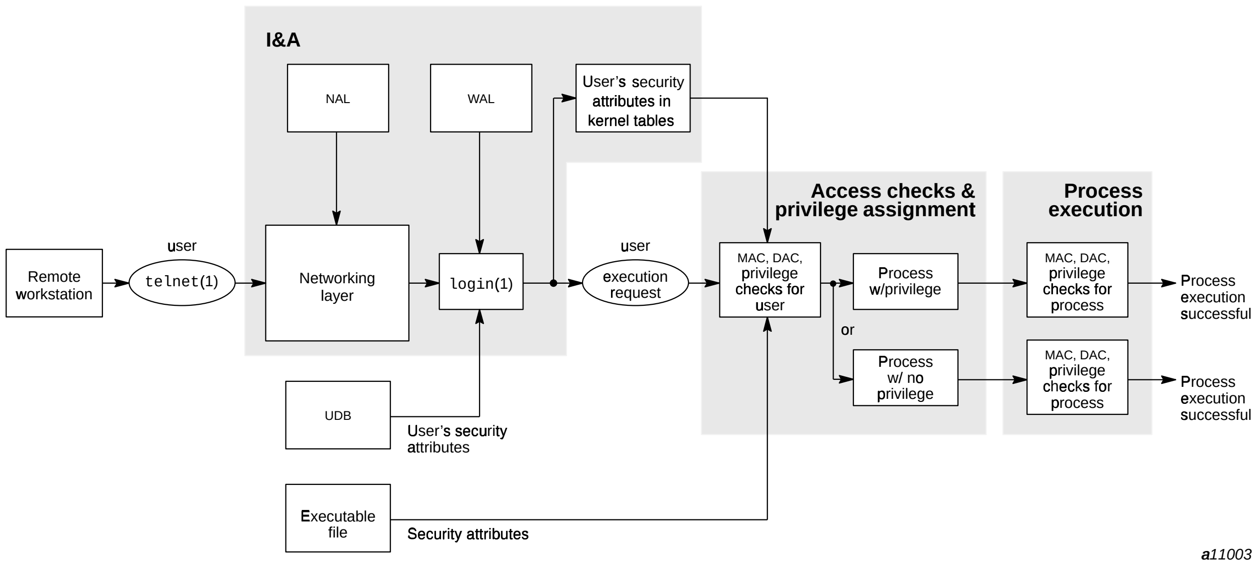
I&A
NAL
WAL
User's security attributes in kernel tables
Access checks & privilege assignment
Process execution
Remote workstation
user
telnet(1)
Networking layer
login(1)
user
execution request
MAC, DAC, privilege checks for user
Process w/privilege
or
Process w/ no privilege
MAC, DAC, privilege checks for process
MAC, DAC, privilege checks for process
Process execution successful
Process execution successful
UDB
User's security attributes
Executable file
Security attributes
a11003

Figure 6. Interaction of UNICOS security mechanisms

Security auditing is not shown in Figure 6 because auditing overlays the whole system. That is, auditing records can be generated by various actions, which are not easily shown in the figure. For an introduction to auditing, see Section 8.8, page 261.

Security object reuse is not shown in Figure 6 because this mechanism is inherent in the UNICOS operating system and is not easily shown in the figure. For an introduction to object reuse, see Section 8.6, page 230.

I&A mechanisms are explained in more detail in Section 8.5, page 204. DAC mechanisms are described in more detail in Section 8.3, page 171. MAC mechanisms are described in more detail in Section 8.4, page 173. System management mechanisms are described in more detail in Section 8.2, page 148.

8.2 System management

Many different administrative tasks are performed by multiple administrators with differing levels of expertise and authority. Operator tasks often differ from those of a security or system administrator. On a traditional UNIX system, the use of `root` (USER ID 0) allows for relatively easy administration of a computer system. `root` can override virtually all system restrictions in order to perform a task, possibly with no way to trace the action back to a specific user. This potential lack of administrative accountability is undesireable.

System management (or trusted facility management (TFMgmt), as it is referred to in the TCSEC) allows administrative work to be accomplished on a UNICOS system. The principal requirements of system management, as defined in *A Guide to Understanding Trusted Facility Management*, publication NCSC-TG-015, are as follows:

- The separation of operator and administrator functions.
- The logical (or physical) separation of the database information corresponding to those functions.
- The implementation of least privilege such that functions have only the minimum necessary privileges to the databases.

On a UNICOS system, system management is supported by the following mechanisms:

- A super-user mechanism
- The privilege assignment list (PAL)-based privilege mechanism

The super-user mechanism (enabled by the `PRIV_SU` configuration parameter) allows the `root` user to override virtually all system restrictions. This mechanism provides the traditional method of system administration support on a UNICOS system.

The PAL-based privilege mechanism uses privilege assignment lists (PALs) to support the *principle of least privilege*, which is the ability to grant each subject the most restrictive set of privileges for only as long as needed to perform a set of authorized tasks.

With the PAL-based privilege mechanism, PALs are associated with files that administrative users typically execute. When a user executes a file, privilege attributes from the PAL are assigned to the resulting process. The assigned privilege attributes can vary, depending on the active category of the user.

A process with no active categories (that is, running on behalf of a nonadministrative user) can also be assigned privilege attributes from the PAL of the file. This provides the traditional set-user-ID-root functionality by allowing nonadministrative users to perform limited administrative functions in a controlled environment.

The PAL-based privilege mechanism is always available and cannot be configured through a configuration option. However, this mechanism is effective only after PALs have been assigned to files by using the `privcmd`(8), `setpal`(8), and/or `setprivs`(8) commands.

⚡ **Warning:** In UNICOS 9.2 and later releases, all sites are required to assign PALs. The supported privilege configurations are as follows:

- PALs augmented by `PRIV_SU`
- PALs only

The following sections provide more information about the super-user mechanism, the PAL-based privilege mechanism, and information about UNICOS categories used by the PAL-based privilege mechanism.

8.2.1 The super-user mechanism (`PRIV_SU`)

The UNICOS operating system supports a fully functional super-user mechanism. This mechanism works as the super user does on earlier UNICOS systems that do not use the MLS feature, and it allows a process with an effective user ID of 0 to override UNICOS restrictions.

The `PRIV_SU` configuration parameter enables the use of the super-user mechanism. To enable or disable this parameter, use the `Configure system->Multilevel security (MLS) configuration->System options->Super-user privilege policy?` selection in the UNICOS Installation and Configuration Menu System. The default setting for this mechanism is `ON`.

8.2.2 UNICOS categories

UNICOS categories are used on systems that use the PAL-based privilege mechanism to identify administrative roles. Each role is represented by a category. A category has a name and a corresponding bit value within a 32-bit mask. The following categories are defined in the `tfm.h` file:

Category	Description
`secadm`	Defines the security administrator role
`sysadm`	Defines the system administrator role
`sysops`	Defines the system operator role
`unicos`	Not currently used/reserved for use by Cray Research

`sysfil`	Not currently used/reserved for use by Cray Research
`archive`	Not currently used/reserved for use by Cray Research
`datamgr`	Defines the data migration daemon
`netadm`	Not currently used/reserved for use by Cray Research
`diagadm`	Defines the diagnostic administrator role
`daemon`	Not currently used/reserved for use by Cray Research
`system`	Defines the single-user mode administrator for systems using the PAL-based privilege mechanism.
`smail`	Not currently used/reserved for use by Cray Research

Note: The category names described in the previous list are reserved for use by Cray Research.

The user responsible for creating or modifying administrative accounts should assign each administrative user an appropriate set of authorized categories in the user database (UDB). The authorized categories define the set of categories available to the administrative user.

An administrative user can activate an authorized category (or categories) by using the `setucat`(1) command. The user's active category identifies his or her current administrative role.

The UNICOS Installation and Configuration Menu System does not support the ability for a site to define local administrative categories. If your site requires categories other than those defined by Cray Research, a manual procedure for defining local administrative categories is available to sites with access to UNICOS source code.

8.2.3 The PAL-based privilege mechanism

The PAL-based privilege mechanism uses categories to define administrative roles and abilities. The following general policy is used by the PAL-based privilege mechanism to define the types of abilities allowed to the administrator roles:

- Users with an active `system` (for single-user mode) or `secadm` category are allowed to override all command restrictions, including security label restrictions.
- Users with an active `sysadm` category are allowed to perform typical system administrator functions, but are usually constrained by security label restrictions.
- Users with an active `sysops` category are limited to performing typical system operator functions.

The administrative abilities that are specific to a command on a system using the PAL-based privilege mechanism can be found in the man page documentation for that command. Administrative abilities allowed by a command can be modified by updating the privilege assignment list (PAL) associated with that command.

Warning: The PAL-only configuration of the UNICOS operating system is highly restrictive, designed to meet the needs of the evaluated configuration. This configuration is only recommended where the evaluated configuration is absolutely required.

The system management policy supported by the PAL-based privilege mechanism does not automatically grant special abilities to users based on their active category. Rather, special abilities are granted according to the effective privileges and privilege text of a process.

A process is assigned privileges and privilege text when the user executes an executable file that has been assigned a PAL. A PAL provides mapping from a user's active category to the privileges and privilege text of the process. In general, the UNICOS kernel grants special abilities based only on the effective privileges of a process, while UNICOS commands grant special abilities based only on the privilege text of a process.

An administrator can use the `setpal`(8) and `setprivs`(8) to set the PAL attributes of a file. Also, the `privcmd`(8) command can be used to assign the MLS attributes (including PALs) to files.

For an overview of how PALs and categories work to enforce the assignment of privileges, see Section 8.2.3.1.

8.2.3.1 Overview of process privilege attributes

The following sections provide an overview of how process privilege attributes that are obtained from the privilege assignment list (PAL) are used.

8.2.3.1.1 The process

When a user executes a file, the resulting process is assigned privilege attributes based on the user's active category and privileges and the PAL of the file. The privilege attributes of the process consist of permitted privileges, effective privileges, and privilege text. *Effective privileges* are checked by the UNICOS kernel and allow a process to override specific restrictions. *Permitted privileges* are the privileges that a process can make effective. *Privilege text* is checked by UNICOS commands and allows a process to perform special actions that are controlled entirely within the command.

8.2.3.1.2 Privileges

The UNICOS operating system has a unique privilege for each ability that is traditionally associated with the user `root`. For example, the `PRIV_DAC_OVERRIDE` privilege overrides the permissions mode and access control list (ACL) protections on any object.

The UNICOS system also has defined privileges to override the UNICOS security restrictions. For example, the `PRIV_MAC_WRITE` privilege overrides the mandatory access controls for writing to an object.

When a process issues a system call to request a restricted action, the UNICOS kernel verifies that a process has the specific effective privilege(s) needed to perform the action. For example, a process may issue the `open`(2) system call to open a file, but is denied discretionary access control (DAC) access by the permissions mode and ACL of the file. For this example, the UNICOS kernel would verify if the process had the `PRIV_DAC_OVERRIDE` privilege. If it does, then the DAC access is granted; if not, the kernel rejects the request to open the file.

8.2.3.1.3 Privilege text

Although most processes rely on checks done by the UNICOS kernel to control privileged operations, there can be times when the kernel cannot completely enforce the desired policy. This is when privilege text is used.

An example of using privilege text can be shown with the `passwd`(1) command. On a traditional super-user system, the `passwd` command allows a user to change any user's password if the user executing the command has the real user ID of `root`. If the user is not `root`, then `passwd` allows the changing of only the current user's password.

On a system using the PAL-based privilege mechanism, the `passwd` command verifies if it is running with the `chgany` privilege text. If it is, the `passwd` command allows the changing of any user's password. If not, the `passwd` allows the changing of only the current user's password. If the super-user mechanism is also being enforced, the `passwd` command also allows real user ID `root` to change any user's password.

Enforcement of, and abilities granted by, a specific privilege text is unique for every command. The `chgany` privilege text can be used in other commands to control restricted actions that are unrelated to changing passwords. Conversely, if another command can be used to change passwords, it can use any privilege text value to enforce the same policy as the `passwd` command.

8.2.3.2 UNICOS security privileges

The UNICOS privileges are a granular representation of traditional super-user abilities that are enforced within the UNICOS kernel. That is, instead of allowing a process with effective user ID 0 to override all UNICOS kernel restrictions, a set of specific privileges have been defined that allow a process to override specific sets of UNICOS kernel restrictions.

A process with the following privileges effective is allowed to perform the action described:

Privilege	Description
`PRIV_ACCT`	The following is allowed: • Allowed to use the `acct`(2) system call, which is used to enable or disable process accounting. • Allowed to use the `dacct`(2) system call, which is used to enable or disable process or daemon accounting. • Allowed to use the `devacct`(2) system call, which is used to control device accounting. • Allowed to use the `wracct`(2) system call, which is used to write accounting records.

`PRIV_ADMIN`

Allowed to perform restricted network-related administrative functions. Also allowed to perform various restricted system administrative functions where other privileges do not apply.

`PRIV_AUDIT_CONTROL`

The following is allowed:

- Allowed to open the audit log device file (`/dev/slog`) by using the `open`(2) system call. This privilege is used by the auditing daemon to manage audit data.
- Allowed to disable kernel auditing for itself by using the `setusrv`(2) system call. This privilege is used by Cray ML-Safe processes to manage their own auditing.
- Allowed to obtain its current auditing state by using the `getusrv`(2) system call. If a process does not have this privilege effective when calling `getusrv`, the current auditing state that is returned is indeterminate. This functionality prevents a process from determining whether its actions are being audited.

`PRIV_AUDIT_WRITE`

Allowed to use the `slgentry`(2) system call, which is used to write data to the audit trail.

`PRIV_CHOWN`

When `{_POSIX_CHOWN_RESTRICTED}` is enabled, a process that uses the `chown`(2) system call is allowed to change the owner of a file and specify a group to which it does not belong.

`PRIV_DAC_OVERRIDE`

Allowed to override the permission bit and access control list (ACL) protections on named objects. This privilege is applicable to system calls that accept path name parameters.

`PRIV_FOWNER`

Allowed to act as the owner of a file. This privilege is applicable to system calls that are used to set file attributes.

`PRIV_FSETID`

Overrides the following restrictions:

- The effective user ID of the calling process must match the file owner when setting the set-user-ID (`S_SUID`) or set-group-ID (`S_SGID`) mode bits on that file.
- The effective group ID or one of the supplementary group IDs of the calling process must match the group ID of the file when setting the set-group-ID (`S_SGID`) mode bits on that file.
- The set-user-ID (`S_SUID`) and set-group-ID (`S_SGID`) mode bits on a file are cleared upon successful return from the `chown`(2) system call.
- When `FSETID_RESTRICT` is enabled, a process cannot create or manipulate set-user-ID or set-group-ID files as allowed on traditional UNICOS systems.

`PRIV_IO`

Allowed to perform restricted tape and disk I/O related functions. This privilege is applicable to system calls that make use of I/O drivers.

`PRIV_KILL`

Allowed to send a signal to a process that it does not own.

`PRIV_LINK_DIR`

Allowed to create or delete a hard link to a directory.

`PRIV_MAC_DOWNGRADE`

The following is allowed:

- Allowed to downgrade the active security label of an object.
- Allowed to relabel a socket as multilevel or "fuzzy."

`PRIV_MAC_READ`

The following is allowed:

- Allowed to override security label protections on an object when attempting to gain read, execute, or search permission to that object.
- After a process obtains read access to an object (for example, through the `open`(2) system call), this privilege ensures that the process may continue to read the contents of that object, even if the security label of that object is modified.

`PRIV_MAC_RELABEL_SUBJECT`

Allowed to set its authorized and/or active MLS attributes to any value.

`PRIV_MAC_UPGRADE`

The following is allowed:

- Allowed to upgrade the active security label of an object.
- Allowed to relabel a socket as multilevel or "fuzzy."

`PRIV_MAC_WRITE`

The following is allowed:

- Allowed to override security label protections on an object when attempting to gain write permission to that object.
- After a process obtains write access to an object (for example, through the `open`(2) system call), this privilege ensures that the process may continue to write the contents of that object, even if the security label of that object is modified.

`PRIV_PAL_KEEP`

Overrides the restriction that, when the contents of a file is modified, all associated file privileges and PAL category records are cleared.

`PRIV_POWNER`

Allowed to act as the owner of a process. This privilege is applicable to system calls that are used to set or retrieve attributes of other processes.

`PRIV_PROC_ACCESS`

Overrides the restriction that a `/proc` file process that is not the calling process cannot be accessed if the `/proc` file process has the `PC_NOCORE` flag is set.

`PRIV_RESOURCE`

Allowed to use system calls that set or retrieve session resource attributes (for example, limits, `nice` values, and so on).

`PRIV_RESTART`

Allowed to create or set the attributes of a restart file.

`PRIV_SETFPRIV`

Allowed to set file privileges and PALs.

`PRIV_SETGID`

Allowed to change its real group ID.

`PRIV_SETUID`

Allowed to change its real or saved user IDs.

`PRIV_SOCKET`

Allowed to access a privileged socket.

`PRIV_TIME`

Allowed to set a time adjustment value for the system clock.

8.2.3.3 Process privileges

A process is allowed to override system restrictions that are enforced by the UNICOS kernel only if it possesses the appropriate granular privilege or privileges that are required to override those restrictions. Every process has two sets of privileges: permitted and effective. These privileges are defined as follows:

Privilege set	Description
Permitted privileges	Privileges that are authorized for use by a process. These privileges do not allow a process to override system restrictions. The permitted

	privileges of a process are a superset of its effective privileges.
Effective privileges	The privileges with which a process is currently functioning. These privileges are checked by the kernel to determine if the process can override system restrictions. The effective privileges of a process are a subset of its permitted privileges.

A process can add or remove any of its permitted privileges to or from its effective privileges. The permitted privileges serve as a base set of privileges that can be made effective. A process cannot add privileges to its permitted privileges, but it can remove privileges.

8.2.3.4 Privilege assignment list (PAL)

A process is assigned permitted and effective privileges when a user executes an executable binary file that has been assigned a privilege assignment list (PAL). A PAL is comprised of file privileges and PAL category records, which are explained in the following sections.

8.2.3.4.1 File privileges

Every executable binary file has three sets of privileges: allowed, forced, and set-effective. When a process executes a executable binary file, the permitted and effective privileges of a process are modified, based on the privileges of the file. If a file has not been explicitly assigned file privileges, the default value for each set of privileges is the null set. File privileges are defined as follows:

Privilege set	Description
Allowed privileges	The maximum set of privileges that are inherited from the process that executed the file. The intersection of the allowed privileges with the permitted privileges of the process becomes the permitted privileges of the new image process.
Forced privileges	These privileges are unconditionally added to the permitted privileges of the new process image.

Set-effective privileges	These privileges are made effective for the new process image. Only the privileges that are also permitted for the process can be made effective.

An administrator can modify the privilege sets of a file by using the `setprivs`(8) command.

8.2.3.4.2 PAL category records

After the permitted and effective privileges of a process have been initialized using the allowed, forced, and set-effective file privileges, the permitted and effective privileges of a process are further constrained based on the user's active category. This refinement is performed by using PAL category records.

A PAL category record is comprised of three components: an active category, privileges, and privilege text. Each record is in the following form:

```
Active_category:Privileges:Privilege_Text
```

When a user's active category matches that of a PAL category record, the permitted and effective privileges of a process are further constrained by the privileges specified in that record. A `PRIV_NULL` privilege in the `Privileges` field indicates that no privileges are assigned to the process.

The privilege text value in the record is also assigned to the process. A `TEXT_NULL` privilege text in the `Privilege_Text` field indicates that no privilege text is assigned to the process.

Every executable binary file has at least one PAL category record. If a file has not been explicitly assigned a PAL category record, the default record is as follows:

```
other:PRIV_NULL:TEXT_NULL
```

If a user's active category does not match any PAL category records, then the `other` PAL category record is used to further constrain privileges and to assign privilege text. If the user has no active category, the `other` PAL category record is used. The privileges and privilege text of the `other` PAL category record can be modified, but the `other` PAL category record cannot be removed.

Warning: Cray Research has defined PALs for executable files in the set of Cray ML-Safe components. If a site modifies a UNICOS Cray-ML Safe component PAL for a file to define a local administrative policy, all PAL category records must be defined in terms of the privileges and privilege texts defined by Cray Research for that file. The site should modify only the active categories specified in the PAL category records and include only the privilege text values as specified on the man page for that file.

Assigning PALs to files that are not part of the set of Cray ML-Safe components, or modifying UNICOS Cray ML-Safe component PALs for files such that a PAL specifies a set of privileges not defined for that file by Cray Research, should not be done on a Cray ML-Safe system configuration.

An administrative user can update the PAL category records of a file by using the `setpal`(8) command.

8.2.3.4.3 Privilege text

Privilege text is assigned to a process through a PAL category record. This text is a character sequence of up to 8 characters in length. Privilege text is associated only with PAL category records that have a specific active category. That is, it is not typically associated with the `other` PAL category record.

Privilege text is used to identify a type of administrative user. For example, a command may grant special abilities to a specific type of administrative user. However, if the command just checked for a specific active category, there would be little flexibility for a site to customize the administrative policy.

Instead, commands can check for a specific privilege text to determine if the user should be granted special abilities. This allows a site to customize the administrative policy by updating the PAL category records with various active categories, but specifying the privilege text that the command checks.

A command uses the `cmptext`(2) system call to check privilege text values. You can use the `privtext`(1) command to display the privilege text that is assigned if a user executes a specific command. See the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011, for examples on using this command.

8.2.3.5 Propagation of privileges

Figure 7 shows how the process privilege state, the file privileges, and the privilege assignment list (PAL) form and propagate privileges across an `exec`(2) system call.

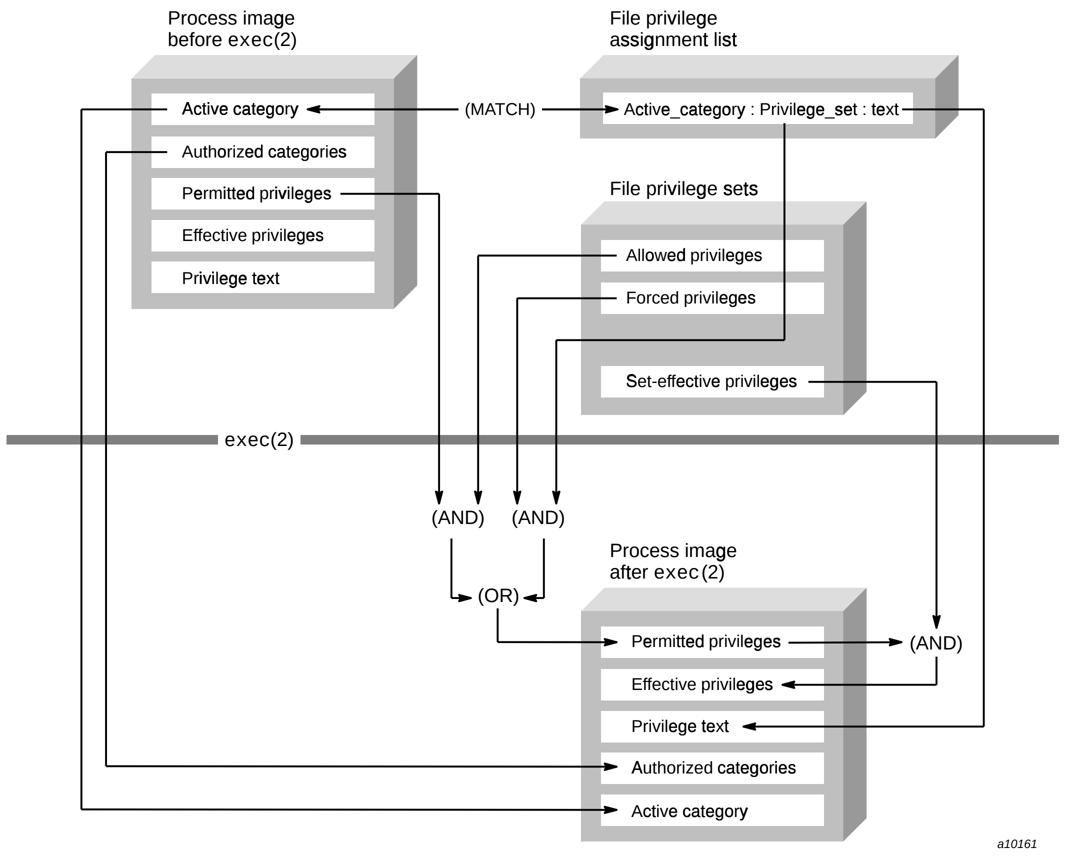

Figure 7. Propagation of privileges

8.2.3.6 Super-user PALs

To make administrative applications function on a system using privilege assignment lists (PAL), changes to application source code are often required. Simply assigning a privilege assignment list to an application does not guarantee that the application will run properly.

Administrative applications often contain internal checks for real and/or effective user ID 0 and can grant special abilities to users with those attributes. Set-user-ID `root` applications often toggle their effective user ID between user ID 0 and the user's real user ID to control the availability of traditional root abilities.

On a system using PALs, dependence on user ID 0 is not desirable because administrative users on such a system do not necessarily run with user ID 0. Applications that check for user ID 0 may not grant administrators the abilities that they require or, in the worst case, the application can simply fail to run.

Modifying application source code to make it work on a system using PALs may not be an option for some Cray sites. Sites may not have the technical understanding of application source code that is necessary to safely modify it or do not always have access to application source code.

The goal of super-user PALs is to provide a relatively easy way for customers to make applications work on a system using PALs without requiring modifications to application source code.

To make an administrative application run on a system using PALs, the application executable binary file must be assigned the `priv_root` flag. This flag can be assigned using the `spset`(1) command.

The file should also be assigned a PAL. To guarantee that the program runs with all traditional `root` abilities, the following file privileges should be assigned:

File privileges	Privilege
Allowed	`PRIV_NULL`
Forced	`PRIV_ALL`
Set-effective	`PRIV_ALL`

The following PAL category record should also be assigned:

```
system:PRIV_ALL:TEXT_NULL
other:PRIV_NULL:TEXT_NULL
```

Additional category entries can also be specified to allow privileged execution by various types of administrators.

When a user executes an executable binary file that has been assigned a PAL and the `priv_root` flag, permitted and effective privileges and privilege text are initially assigned to the process according to the usual privilege assignment algorithm. However, once permitted and effective privileges have been assigned, they propagate across subsequent program executions and are not constrained by PALs.

Also, the real and saved user IDs of the process are forced to 0. If the file is a set-user-ID file, the effective user ID of the process is initialized to the file owner. Otherwise, the effective user ID of the process is forced to 0.

Forcing user IDs to 0 is necessary to address potential user ID 0 checks within the application. This means that when an administrator executes such a file, the process runs as if it were executed directly by the `root` user regardless of the administrator's original user IDs.

If a process sets the value of its effective user ID to 0, all of the permitted privileges of the process are made effective. If a process sets the value of its effective user ID to nonzero, the effective privileges of the process are cleared.

If a process uses the `setuid`(2) system call to change its real, effective, and saved user IDs to nonzero values (that is, removes its ability to function as the `root` user), the permitted and effective privileges of the process are automatically cleared.

8.2.3.7 Software not part of the set of Cray ML-Safe components

Cray Research has defined PALs for executable files in the set of Cray ML-Safe components. UNICOS commands that are not included in the set of Cray ML-Safe components may not have PALs defined by Cray Research. Such commands may be required for administrative tasks, but will not function in a strict PAL-based privilege environment because PALs (or super-user PALs) have not been assigned.

The UNICOS Installation and Configuration Menu System is assigned a super-user PAL and allows escaping to a specially-privileged shell. By escaping to this shell and running an administrative command that is not included in the set of Cray ML-Safe components, it allows the command to function properly in a strict PAL-based privilege environment.

Warning: Using the specially-privileged shell in multiuser mode to run commands that are not included in the set of Cray ML-Safe components should not be done on a Cray ML-Safe system configuration. The specially-privileged shell should be used only from single-user mode.

8.2.3.8 Determining PAL privileges

To determine the privileges to include in the PAL for a command, the command should be analyzed by investigating the code and associated documentation. The following example shows how the `cat`(1) command can be analyzed.

The `cat` command displays file contents. This involves reading data from one or more files and writing that information to standard output. The `cat` command depends only on the UNICOS kernel to grant special abilities to administrative users.

For administrative users, the `cat` command (through the UNICOS kernel) allows reading data from any file and writing that information to standard output (which can be redirected to any file). The privileges involved in overriding the security label, permission bit, and access control list (ACL) protections of any file are the `PRIV_MAC_READ`, `PRIV_MAC_WRITE`, and `PRIV_DAC_OVERRIDE` privileges.

If you do not completely understand the high-level functionality of a command, you can construct a set of required privileges by determining the privileges that are used in each system call that the command directly or indirectly invokes. The man pages for the system calls contain the privileges associated with each system call.

The `cat` command does not internally alter its behavior based on any user attributes (for example, user ID, active category, and so on). This means that only the `TEXT_NULL` privilege text is required.

The `cat` command should not grant special abilities through the UNICOS kernel to nonadministrative users. This means that the `PRIV_NULL` privilege should be associated with nonadministrative users, which is done as follows:

```
other:PRIV_NULL:TEXT_NULL
```

The default administrative policy for PAL-based privilege ensures that users with an active `system` or `secadm` category are allowed to override all mandatory access and discretionary access control protections of a file. This means the PAL category records for the `system` and `secadm` categories should contain the `PRIV_MAC_READ`, `PRIV_MAC_WRITE`, and `PRIV_DAC_OVERRIDE` privileges.

The default administrative policy for PAL-based privilege ensures that users with an active `sysadm` category can override only the discretionary access control protections on a file. This means the PAL category record for the `sysadm` category should contain only the `PRIV_DAC_OVERRIDE` privilege.

This results in the following PAL category records:

```
system:PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE:TEXT_NULL
secadm:PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE:TEXT_NULL
sysadm:PRIV_DAC_OVERRIDE:TEXT_NULL
other:PRIV_NULL:TEXT_NULL
```

Defining the allowed, forced, and set-effective file privileges can be based on the privileges in the PAL category records. Privileges are rarely, if ever, inherited from a previous process. This means the allowed privileges contain only `PRIV_NULL`. The forced and set-effective privileges contain all the privileges specified in the PAL category records. This means the file privileges consist of the following:

```
allowed = PRIV_NULL
forced = PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE
set-effective = PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE
```

8.2.3.9 Process privilege management

A process should begin execution with the minimum set of permitted privileges that are necessary to complete its task. Some or all of those privileges may also be effective upon execution using the set-effective privileges of the executable file.

A process does not have to remove effective privileges if leaving those privileges effective for the life of the process does not cause the process to perform an action that compromises the system security policy. If a process must perform an action that, because one or more privileges are effective, causes the process to perform an action that places the system security policy at risk, the process should remove the undesired effective privileges.

If a process must execute a file, the process should remove privileges that are not required for propagation from its permitted (and effective) privileges.

UNICOS library routines allow a process to set, retrieve, and manipulate its permitted and effective privileges. The names of these routines have the prefix `priv_`. See the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080, for more information on these routines.

8.2.3.10 Privilege text management

A program that internally verifies user IDs and/or categories does so by retrieving these attributes with the `getuid`(2), `geteuid`(2), and `getusrv`(2) system calls. The retrieved information is then compared against a desired value (for example, user ID 0, `secadm` category, `sysadm` category, and so on) and the program alters its behavior accordingly.

The altered behavior may consist of granting special abilities to a user who possesses specific attributes and granting more restrictive abilities to all other users. For example, on UNICOS 7.0 MLS systems, the `/bin/passwd` command allows a user with real user ID 0 to modify any user's password, but restricts all other users to modifying only their own password. The `/bin/passwd` program accomplishes the check for real user ID 0 using the `getuid` system call.

In a PAL-based privilege environment, administrative tasks are not performed by users with user ID 0. Instead, programs grant special abilities based on the privilege text. Programs use the `cmptext`(2) system call to check privilege text values. The `cmptext` system can be used to verify if a user has effective user ID 0, real user ID 0, or a specific privilege text value. This system call is used as a replacement for the `getuid` and `geteuid` system calls to determine if a user should be granted special abilities.

8.2.4 Privileged shell

The UNICOS user environment relies on the shell for command execution, I/O redirection, and management of a user's session. Proper execution of the shell requires that an administrator set up the necessary conditions for the correct execution of commands using the shell. The administrative work may include changing the working directory, redirecting I/O, setting the label of the process, and so on.

Note: On the UNICOS system, the default shell (`/bin/sh`) is the Korn shell.

To make it possible for administrators to execute these administrative functions correctly on a UNICOS system, privilege is needed in some cases. The privileged shell is used on UNICOS systems to provide the mechanism needed to enable and manage privileges within the shell when appropriate. The privileged shell is available only if the system is using the PAL-based privilege mechanism.

This privileged shell can be used to accomplish the following tasks by the security administrator, system administrator, and system operator roles:

- The security administrator can override all MAC and DAC restrictions on files and directories, as well as override all restrictions on killing processes, setting the shell process label, and changing the shell process resource restrictions. The following shell built-in actions are privileged for the security administrator:

  - I/O redirection
  - File name expansion
  - `cd`(1)
  - `echo`(1)
  - `kill`(1)
  - `print` (see `ksh`(1))
  - `pwd`(1)
  - `read`(1)
  - `setusrv`(1)
  - `setucmp`(1)
  - `setulvl`(1)
  - `test`(1)
  - `ulimit` (see `ksh`(1))

- The system administrator can override DAC restrictions on files and directories, kill processes regardless of process ownership (subject to the MAC restrictions), and override restrictions on changing the shell process resource limits. The system administrator does not have the authority to override MAC restrictions or set process labels. The following shell built-in actions are privileged for the system administrator:
  - I/O redirection
  - File name expansion
  - `cd`(1)
  - `kill`(1)
  - `pwd`(1)
  - `test`(1)
  - `ulimit` (see `ksh`(1))
- The operator can kill processes regardless of process ownership (subject to the MAC restrictions) and override restrictions on changing the shell process resource limits. The operator does not have the authority to override MAC

restrictions, override DAC on files, or set process labels. The following shell built-in actions are privileged for the operator:

- `kill`(1)
- `ulimit` (see `ksh`(1))

To use the privileged shell, an administrator enters the shell by activating the appropriate administrative or operator category (by using the `setucat`(1) command) and executing a subshell.

8.2.5 Overview of access and privilege checks

This section shows the relationship between mandatory access control (MAC), discretionary access control (DAC), and privilege checks by describing a nonadministrative user login and command execution sequence.

After `login`(1) successfully sets up the kernel table MLS attributes for the user' s `login` shell, the user is free to execute a command. As shown in , this results in a new child process (which is essentially a mirror image of the `login` shell).

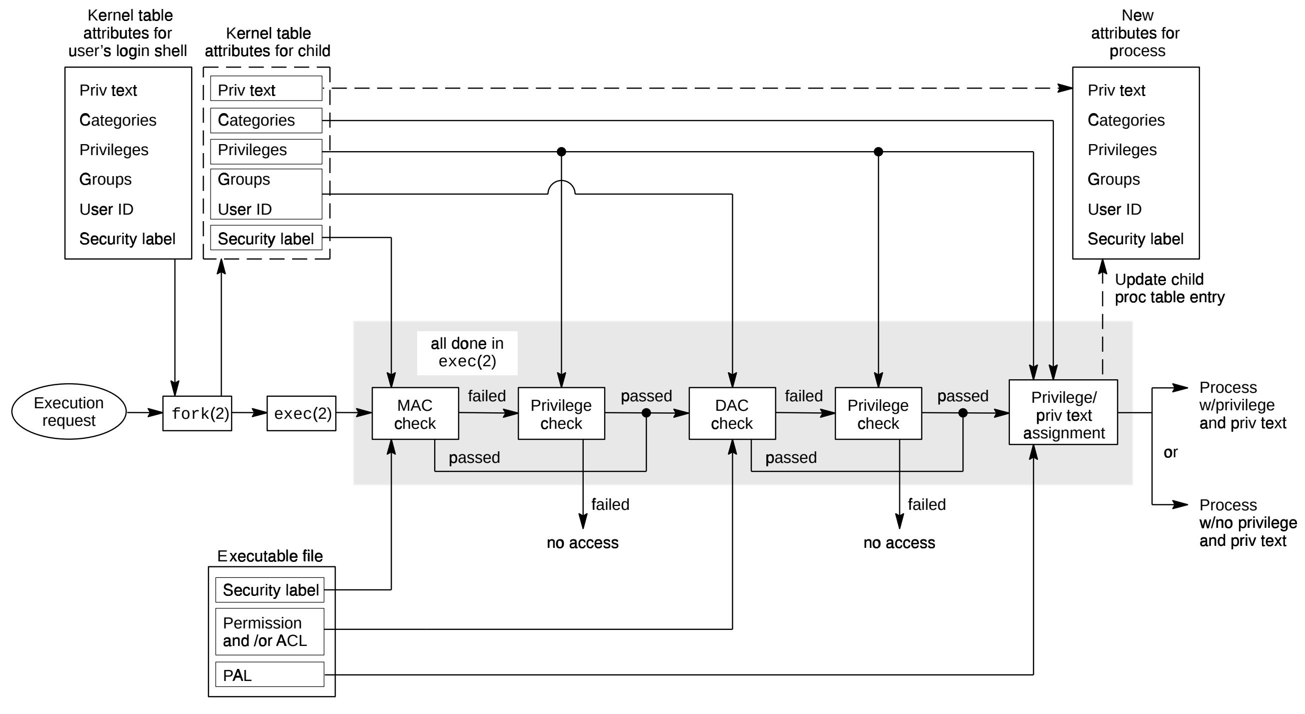

Figure 8. Overview of initial MAC/DAC checks and assigning of privileges

The `exec`(2) system call then verifies that the child process has access to the command.

MAC access is granted if either of the following conditions are true:

- The active security label of the child process dominates the security label of the command
- The child process has the appropriate privilege to override the MAC policy for file execution

If neither condition is true, then execute permission to the file is denied.

If either condition is true, then DAC checks are performed. The group IDs and effective user ID assigned to the login shell are compared against the permisssion bits and ACL of the executable file, as defined by the algorithm outlined in the *UNICOS Multilevel Security (MLS) Feature User's Guide*, Cray Research publication SG–2111.

If DAC access is granted, the process is assigned permitted privileges, effective privileges, and privilege text based on the user's active category and the PAL of the file. The effective user and group IDs of the process could change if the file was a `setuid` or `setgid` program.

8.3 Discretionary access control

Discretionary access control (DAC) is implemented through a set of rules that control and limit access to an object, based on an identified individual's need to know and without intervention by a security officer for each individual assignment.

This is accomplished by the use of standard UNICOS mode permission bits and an access control list (ACL); the ACL and mode bits allow the owner of a file to control read (r), write (w), and execute (x) access to that file. The file's owner can create or modify an ACL that contains the identifiers and the r/w/x permissions for those individuals and groups that are allowed to access the file.

An ACL contains one or more ACL entries; each ACL entry defines file access information for a user. The entry contains a user field (which defines the user's login name), a group field (which defines the group name), and the permissions field (which is used to define any combination of r/w/x or define no (n) access).

The ACL entries, which define the absolute permissions, are intersected with the file's group (mask) bits to determine the type of discretionary access allowed; this is called the effective permissions.

Object access is always governed by the mandatory policy restrictions established by the security administrator.

Refer to the section on using ACLs in the *UNICOS Multilevel Security (MLS) Feature User's Guide*, Cray Research publication SG–2111, for more information on the following:

- How ACLs are used
- Examples of how to create and maintain ACLs

You can also refer to `spacl`(1) in the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011.

8.3.1 `umask` on a MLS system

On a Cray ML-Safe system configuration, the default setting for `umask`(1) is 077. This default is defined in the `umask` command in `skl/c1/etc/profile` and `skl/c1/etc/cshrc`.

This default should not be changed on a Cray ML-Safe system configuration. Users of `umask` are affected in the following ways:

- Users that rely on new files automatically having group and/or world access must manually set the file access permission bits to enable this access.
- Users who want a `umask` other than 077 must place the `umask` command in their `$HOME/.profile` or `$HOME/.login` file.

Site administrators should ensure that any affected users are made aware of these changes.

8.3.2 Managing set-user-ID and set-group-ID files

The UNICOS set-user-ID (setuid) or set-group-ID (setgid) functionality can be very useful, although it poses some security risks. A poorly designed setuid or setgid program can compromise the security of the owner or owning group of a file. When the owner of a file grants access permission to other users, the owner effectively relinquishes control over the setuid or setgid file.

Previous releases of the UNICOS system with the MLS feature enabled have restricted the management of setuid and setgid files. The UNICOS system uses the `FSETID_RESTRICT` configuration parameter to allow sites to restore the UNICOS non-MLS functionality of setuid and setgid files.

When `FSETID_RESTRICT` is enabled, only appropriately authorized users can create or link to setuid or setgid files. Only appropriately authorized users can open setuid or setgid files for writing. Also, for `chmod`(2), `chown`(2), and for setting access control list (ACL) operations, the setuid and setgid mode bits of a file are cleared if the user is not properly authorized.

When `FSETID_RESTRICT` is not enabled, the UNICOS (POSIX) policy on setuid and setgid files is enforced. In addition, when setting an ACL on a file, the setgid mode bit of a file is cleared if the user does not belong to the owning group of the file and is not an appropriately authorized administrative user. This functionality makes the set-ACL behavior consistent with the behavior of `chmod`.

The `FSETID_RESTRICT` parameter is enabled by default. To enable or disable this parameter, use the `Configure system->Multilevel security (MLS) configuration->System options->Restrict setuid/setgid file creation` selection in the UNICOS Installation and Configuration Menu System.

Note: For the UNICOS 10.0 release, the default value of the `FSETID_RESTRICT` configuration parameter will be changed to `OFF`.

The `spcheck`(8) utility allows the security administrator to search the system for setuid and setgid files and to maintain surveillance of their use. See Section 8.8.9, page 352, for more information.

8.4 Mandatory access control

Mandatory access control (MAC) is implemented through the UNICOS security policy, which is a set of rules that control access based directly on a comparison of the subject's clearance and the object's classification. The UNICOS security policy is enforced for all attempts to access an object.

The security policy controls read and write operations in order to prohibit unauthorized disclosure of any system or user information. The security policy is defined as the set of rules and practices by which a system regulates the disclosure of information. The mandatory security policy enforced by the UNICOS MLS feature is as follows:

- A subject may read or execute an object only if the active security label of the subject dominates the security label of the object.
- A subject may write to an object only if the security label of the subject is equal to the security label of the object.

A security label consists of a security level and zero or more security compartments. A maximum of 17 hierarchical security levels and 63 nonhierarchical compartments are used to represent a nonadministrative subject's clearance and an object's classification on the UNICOS system.

The security levels can range from through 16, with having the lowest value and 16 the highest. The UNICOS system also uses system high (`syshigh`) and system low (`syslow`) labels. See Section 8.4.2.1, page 186, for more information on these labels.

An appropriately authorized administrator can use the `nu`(8) or `udgben`(8) commands to assign each user a minimum, maximum, and default security level in the user database (UDB). The security administrator also assigns each user a set of active and authorized compartments in the UDB. The active compartments define the set of compartments with which a user is currently functioning. The authorized compartments define the range of comparments that a user can add to his or her active set of compartments. For more information on how these values are used to determine a user's security attributes, see the *UNICOS Multilevel Security (MLS) Feature User's Guide*, Cray Research publication SG–2111.

Objects are assigned security labels by the installation process, administrative procedures, or by inheriting the active security label of the subject who creates it. If an object is not explicity assigned a security label by one of these methods, it has a security label of level 0 and null compartments by default.

A session's security attributes are set at session initiation. These attributes are deterrmined by the information defined in the UDB plus information defined for network and other configuration options.

MAC checks are always performed before discretionary access control (DAC) checks. If either of the conditions in the first two bullets described previously are met, then DAC checks are made. Otherwise, the subject is denied access to the object.

The security administrator can use the `deflbl_as_minlbl` configuration field `/etc/config/confval` to allow minimum security labels to be defined for users. Enabling this field allows the user's default security label (that is the `deflvl` and `defcomp` fields in the user database) to become the user's minimum security label. Use of this field means users must log in with a security label that dominates their default security label.

The security administrator can also use the `mincomps` field in the UDB to define a user's minimum compartment set. See Section 8.5.3, page 208, for more information on these fields.

The security administrator can also define a set of permissions for each user in the UDB. These permissions grant special privileges to the user and are defined in the security parameter file (`sys/secparm.h`). See Section 8.7.5.4, page 237, for more information on these permissions.

For more information on how these values are used to determine a user's security attributes, see the *UNICOS Multilevel Security (MLS) Feature User's Guide*, Cray Research publication SG–2111. This manual explains the interactive login process for both UNICOS and Cray ML-Safe system configurations. For more information on setting up UDB security entries, see Section 8.7.6, page 239, for more information.

The following sections describe how the UNICOS MAC policy affects the following system operations:

- Directory operations
- File system operations
- Device labeling
- `cron` and `at` operations
- Cray ML-Safe mail
- `/proc` operations
- `syslogd` operations
- IPC objects

8.4.1 Directory operations

You can use the `mkdir -L` command to create a directory with a security label that is different than your active security label (assuming the requested label is within your authorized range). If the relabeling fails, the directory's label remains at your active security label. If the `mkdir -L -p` command is used, only the last directory in the path is relabeled. All intermediate directories are created at your active label.

Any user can upgrade the label of an empty directory (for example, when using the `spset -c` or `spset -l` commands) if all of the following conditions are met:

- The user has MAC write access to the target directory
- The user is the owner of the target directory
- The target directory is empty

If you are properly authorized, you can override these restrictions and change the label of any directory; the definition of properly authorized depends on which system management mechanism your system is using. For more information, see the `mkdir`(1) man page in the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011.

To create a directory, the security label of the directory must always fall within the security label range of the file system on which the directory resides.

8.4.1.1 Removing files from directories

To ensure compliance with the TCSEC object reuse requirements, the UNICOS system clears the name of the object being removed from the directory. Only the object name is cleared by this change. The remaining entries in the directory entry structure (for example, inode number, name signature, record length and name length) are not cleared.

Administrators should be aware that directories with removed objects that exist on UNICOS 8.0 MLS systems continue to have "removed" object names in them on later UNICOS releases until the directory entries are replaced with another object or the directory block is released.

8.4.1.2 Wildcard and multilevel directories (MLDs)

The UNICOS MLS feature uses two mechanisms for labeling directories that contain objects at different security labels: wildcard directories and multilevel directories (MLDs). The following sections explain the function and use of these directories. These two types of directories can be used exclusively or in combination on a UNICOS system. Wildcard directories should not be used on evaluated configurations of the UNICOS operating system. Cray Research recommends using MLDs for products that process multiple security labels (for example, `mail`, `lpr`, `lpd`, `cron`, NQS, and CRL). See Section 8.4.1.2.2, page 177, for more information.

8.4.1.2.1 Wildcard directories

The UNICOS MLS feature uses security level 63 to indicate a wildcard directory. A wildcard directory can contain files at any security label within the boundaries of the file system. Access to files in the wildcard directory must satisfy the discretionary and mandatory access policy enforced by the UNICOS MLS feature.

You can apply security compartments to a wildcard directory, although these compartments are ignored by the system when doing a mandatory access control check.

Wildcard directories are established primarily for daemons that must service many requests and output queues; the assignment of wildcard directories should be restricted to those described in Section 8.7.7, page 241. Also, the use of wildcard directories avoid replication of special directories for every use.

The `/usr/tmp` and `/tmp` directories, which are accessible to many system utilities, must be assigned the wildcard security level (or converted to multilevel directories (MLDs), which is explained in the next section) to allow them to contain files with varying security levels.

The Network Queuing System (NQS) spooled output and data files directories must also be labeled as wildcard directories (or MLDs). The NQS directories are automatically labeled by NQS.

For more information on labeling these directories, see Section 8.7.7, page 241.

8.4.1.2.2 Multilevel directories (MLDs)

The UNICOS system has relied upon the use of wildcard directories to hold files at multiple labels. These directories have been used for public directories like `/tmp` as well as for spool directories (for example, `/usr/spool/mqueue`).

The use of wildcard directories violates the TCSEC requirements, as they create a potential for write-down security policy violations. Wildcard directories have been replaced with MLDs.

Warning: Only the use of MLDs is allowed on the evaluated configuration of a UNICOS system. There are no checks or warnings enforced by an evaluated system to ensure only MLDs are used. This requirement must be enforced by administrative procedures. There is no configuration parameter in the UNICOS Installation and Configuration Menu System that enables MLDs.

MLDs provide a method of sharing a common directory name, while partitioning the actual directory contents according to security labels.

For example, a wildcard directory provides a single name space at all labels. Because of this, if `/tmp` is labeled as a wildcard directory, a process at level 0 and compartment set 077 can create `/tmp/file` at level 0 and compartment set 077, but a process at level 0 and compartment 0 would be unable to create `/tmp/file` at level 0 and compartment set 0, as that file already exists in the directory and is not writable by the second process.

A MLD has two parts: a root directory and a multilevel symbolic link to the root directory. A MLD provides a discrete name space for each label represented within the directory. As a result, if `/tmp` is a MLD, a process at level 0 and compartment set 077, and a process at level 0 and compartment set 0 can each create a file known to each process as `/tmp/file`. The result is two different files at two different labels containing completely different data.

Warning: Because of the nature of the symbolic link expansion, MLDs do not work across NFS mount points. It is not recommended to use MLDs on NFS mounted file systems.

The following list contains the Cray Research products or commands that use MLDs when running on a Cray ML-Safe system configuration. The mail directories require the use of MLDs on either a UNICOS or a Cray ML-Safe system configuration:

- `jtmp` directories
- `/tmp` directory
- `/usr/tmp` directory
- `cron`(8) and `at`(1) spool directories
- `lpr`(1) and `lpd`(8) spool directories
- Mail directories (`/usr/mail` and `/usr/spool/mqueue`), plus the user's directories that are used to save mail messages.
- NQS spool directories
- CRL debug log directory as defined by `${RLLOGDIR}`

The following information on MLDs assumes you have read and understood the MLD section in the *UNICOS Multilevel Security (MLS) Feature User's Guide*, Cray Research publication SG–2111.

8.4.1.2.3 Creating a MLD

There are several ways to create a MLD, which are as follows:

- Create a new MLD
- Customize the directory by first creating it and then creating a link to it.
- Convert an existing directory structure to a MLD.

To create a new MLD, use the `mlmkdir`(8) command. This command creates the directory and the symbolic link, labeling both with a mode of `rwxrwxrwx`. This labeling scheme is convenient if you want to create a directory that is available to public use or when creating a directory for users (for example, a user's mailbox directory). A more appropriate mode should be assigned after creating the directory.

The `mlmkdir` command creates the multilevel symbolic link with the name specified on the command line and adds the `.mld` suffix to the name to create the directory. This naming convention is convenient, but not required, as shown in the following example.

The following example shows how to use the `mlmkdir` command. The `ls -l` command is used in this example to show the result of the operation; for ease of use, the output from the `ls` command has been edited to remove the link counts, owners, groups, and creation time:

```
$ setucat secadm
secadm$ pwd
/home/user
secadm$ mlmkdir /home/user/mld
secadm$ ls -l
total 32
drwxr-xr-x  .
drwxr-xr-x  ..
lrwxrwxrwx mld -> /home/user/mld.mld
drwxrwxrwx mld.mld
secadm$ cd mld
secadm$ /bin/pwd
/home/user/mld.mld/000
secadm$ cd ..
secadm$ /bin/pwd
/home/user
secadm$ setucat 0
$
```

If you do not want to use the `.mld` suffix for the directory name, or you want to create a multilevel symbolic link to an existing directory, you can use the `-m` option of the `ln`(1) command. This option works like the `-s` option of the `ln` command, but instead of creating a regular symbolic link, it creates a multilevel symbolic link.

The following example show how to change the `.mld` suffix to a `.dir` suffix. In this example, the directory is created by using the `mkdir`(1) command and then the multilevel symbolic link is created using the `ln` command:

```
$ setucat secadm
secadm$ pwd
/home/user
secadm$ mkdir mymld.dir
secadm$ ln -m /home/user/mymld.dir mymld
secadm$ ls -l
total 48
drwxr-xr-x  .
drwxr-xr-x  ..
lrwxrwxrwx mld -> /home/user/mld.mld
drwxrwxrwx mld.mld
lrwxrwxrwx mymld -> /home/user/mymld.dir
drwxr-xr-x mymld.dir
secadm$ cd mymld
secadm$ /bin/pwd
/home/user/mymld.dir/000
secadm$ cd ..
secadm$ /bin/pwd
/home/user
secadm$ setucat 0
$
```

To remove a MLD, you can use the `mlrmdir`(8) command. This assumes that the directory is empty (that is, only the root directory and labeled subdirectories are present, with no files or directories created by users). The following example removes both directories created in the two previous examples. Use of this command removes both the multilevel symbolic link and the directory:

```
$ setucat secadm
secadm$ pwd
/home/user
secadm$ mlrmdir mld mymld
secadm$ ls -l
total 16
drwxr-xr-x  .
drwxr-xr-x  ..
secadm$ setucat 0
$
```

The `cvtmldir`(8) command provides the ability to convert between wildcard directory and MLD structures.

The `cvtmldir` command allows an administrator to convert a wildcard directory to a MLD, while preserving all files and directory tree structure, and placing each file correctly in the multilevel directory tree.

The `cvtmldir` command also allows an administrator to convert a multilevel directory to a wildcard labeled directory, but the conversion may not be perfect, as file name collisions can result.

To use `cvtmldir` on a `PRIV_SU` system, you must be the super user. To use `cvtmldir` on a system using only the privilege mechanism, you must have an active `system` category.

On a Cray ML-Safe system configuration, the only way to obtain an active `system` category is to bring the system to single-user mode. On systems that use the PAL-based privilege mechanism, but are not strict Cray ML-Safe environments, it is possible that a user be assigned the `system` category in a multiuser state.

The following sections provide examples of converting directories.

8.4.1.2.4 Converting from wildcard directory to MLD

Conversion from a wildcard directory to a MLD can be done by one of the two following procedures:

- As a copy from one directory tree to another
- As a conversion "in-place", using the source directory as the root of the directory tree that is created for the destination

In either case, `cvtmldir` does not change the source directory structure. This avoids loss of data in the event of an unsuccessful conversion. Once conversion completes successfully, you must remove the source directory tree. In either case, `cvtmldir` uses file system links whenever possible to reduce the amount of storage needed to perform the conversion.

8.4.1.2.5 Conversion by copying

If the directory to be converted is not the root of a mounted file system, the simplest way to convert it is by copying from the original directory structure to a new directory structure. In this case, to clean up after the conversion, remove the old directory structure.

For this conversion method, rename the source directory and convert the renamed source directory into a directory with the original name. This reduces the probability that the source directory changes during the conversion and there is no need to rename the destination directory.

The following example shows the conversion of /usr/spool/mqueue on a system with the PRIV_SU configuration option enabled. This example assumes that /usr/spool/mqueue is not the root of a mounted file system and the file system on which /usr/spool/mqueue resides does not support the syslow label.

```
mv /usr/spool/mqueue /usr/spool/mqueue.old
cvtmldir -m /usr/spool/mqueue.old /usr/spool/mqueue
rm -rf /usr/spool/mqueue.old
spset -l 0 /usr/spool/mqueue.mld
```

If your file systems support the syslow security label, then use the spset -l syslow /usr/spool/mqueue.mld command instead of the spset command sequence shown in the previous example.

The following example shows the conversion of /usr/spool/mqueue on a system using only the privilege mechanism (again, this example assumes that /usr/spool/mqueue is not the root of a mounted file system):

```
setucat system
mv /usr/spool/mqueue /usr/spool/mqueue.old
cvtmldir -m /usr/spool/mqueue.old /usr/spool/mqueue
spset -l syslow /usr/spool/mqueue.mld
setucat 0
```

When the conversion is completed, you can remove the /usr/spool/mqueue.old directory.

In the previous example, the setucat system command activates the system category. If this category is already active, ignore this step. The setucat 0 command deactivates the system category. If you are doing other tasks that require the system category immediately after this task, ignore this step.

In all conversion examples in the following sections, the procedures are shown for a system with PRIV_SU enabled. Add the setucat commands when necessary to use the procedures on a system with the PAL-based privilege mechanism.

8.4.1.2.6 Converting in-place

If you want to convert the mount point of a file system into a MLD, you cannot convert by copying as shown in the previous section, as there is no way to make the copy become the root of the file system. To make this type of conversion, you need to convert the directory "in-place". The in-place conversion of a wildcard directory results in the entire wildcard directory structure being located in `/tmp.mld`.

The most reliable way to successfully complete this type of conversion is to do it in single-user mode. This is necessary to unmount and remount the new directory, as `/tmp` is usually busy on a multiuser system.

The `/dev/dsk/tmp` device is used in the following example; if the device containing the `/tmp` file system has a different name on your system, use that device.

The following example shows the conversion of the `/tmp` directory from a wildcard directory to a MLD.

```
umount /dev/dsk/tmp
mv /tmp /tmp.mld
mount /dev/dsk/tmp /tmp.mld
cvtmldir -f -m /tmp.mld /tmp
spset -l syslow /tmp.mld
```

When the conversion is completed, `/tmp.mld` contains a set of directories, each of which has a name of octal digits that begins with a zero. Each of these directories is a labeled subdirectory; the name of each directory is an octal value with the first three characters representing the security level and all following characters representing the octal compartment mask (this naming convention is explained later).

Before `/tmp` can be used as a MLD, remove the entries that are not names of labeled subdirectories from `/tmp.mld`, as follows:

1. Remove all entries that have nonoctal characters in their names or entries that do not consist of at least three octal characters starting with 0, as shown in the following example:

   ```
   cd /tmp.mld
   ls -l | grep -v "0[0-7][0-7]*" | xargs rm -rf
   ```

2. Execute the `ls -l` command. Remove any entry that is not a directory.

3. Examine each remaining directory to determine whether the name of the directory matches the label. To do this, use the `spget`(1) command to obtain the level and compartments of each directory and convert the level value to octal using the following table:

Label	Octal representation
0	000
1	001
2	002
3	003
4	004
5	005
6	006
7	007
8	010
9	011
10	012
11	013
12	014
13	015
14	016
15	017
16	020
51	063 (`syslow`)
54	066 (`syshigh`)
63	077 (wildcard)

Note: The naming convention for MLDs outlined in this section may change in future UNICOS releases.

After converting the level to octal, delete the leading zero on the compartment bit mask and put the two octal values together as follows:

```
<octal level><compartment bit mask>
```

For example, if a directory has a level of 14 and a compartment set of 0705, its name as a labeled subdirectory would be `016705`. If an existing directory name does not match the name you have determined, then it is not a labeled subdirectory and should be removed.

8.4.1.2.7 Converting from MLD to wildcard directory

Warning: Only MLDs are supported on a Cray ML-Safe system configuration.

Converting from a MLD to a wildcard directory can result in name collisions. The `cvtmldir`(8) command renames the colliding file and reports both the old and new name to the administrator. The administrator should tell the affected users that the files are renamed.

Conversion from a MLD to a wildcard directory can be done by one of the two following procedures:

- As a copy from one directory tree to another
- As a conversion "in-place", using the source directory as the root of the directory tree that is created for the destination

The following example shows how to convert a directory that is not the root of a file system to a wildcard directory:

```
rm /usr/spool/mqueue
cvtmldir -w /usr/spool/mqueue.mld /usr/spool/mqueue
```

Successful completion of this example results in a wildcard directory tree in `/usr/spool/mqueue` and a MLD tree in `/usr/spool/mqueue.mld`. Remove the multilevel directory tree as follows:

```
rm /usr/spool/mqueue.mld
```

To convert a MLD that is the root of a file system to a wildcard directory, you must convert it in-place. You may find it useful to know what labeled subdirectories are in the MLD before starting the conversion.

Do this by changing to the directory containing the MLD structure (usually `<pathname>.mld`) and executing the `ls`(1) command to obtain a list of all directories in the root of the MLD. You may want to save the output from the `ls` command execution in a file for future reference.

The following example shows the in-place conversion of `/tmp` from a MLD to a wildcard directory:

```
cd /tmp.mld
ls
cd /
rm /tmp
```

```
cvtmldir -f -w /tmp.mld /tmp.mld
umount /tmp.mld
mv /tmp.mld /tmp
mount /tmp
```

Successful execution of this conversion results in a wildcard directory representation of the MLD structure, although there may be residual labeled subdirectories. These are the directories in the list obtained from `ls` before the conversion. Remove these directories.

8.4.1.3 Directory permissions

Regardless of whether the UNICOS MLS feature is enabled, directories such as `/bin` and `/etc` should not have public write access permission assigned to them. If public write permission exists on such a directory, a user could conceivably replace an existing command with a modified version with the same name, thus introducing a Trojan horse.

A suitable access mode for these directories is 755, which specifies public read access. Read permission allows users to read a directory as a file, discovering all the names it contains. For similar reasons, a user's home directory and `.profile` (see `sh`(1)) or `.login` and `.cshrc` (see `csh`(1)) files should be owned by that user, and write access should be restricted to the file owner.

8.4.2 File system and file operations

The following sections explain the labeling and use of file systems and files on a UNICOS system by providing the following information:

- System high and system low labels
- File system labeling
- File system access control
- File system backup
- File system security
- File labeling

8.4.2.1 System high and system low labels

The ability to protect system files from unauthorized access or modification was provided on pre-8.0 MLS systems by discretionary access controls (DAC) only.

In order to support the UNICOS security policy and TCSEC criteria, the UNICOS system now uses the system high (`syshigh`) and system low (`syslow`) security labels.

Only a properly authorized user can override these labels to modify, read, or write to a system file. The `syshigh` and `syslow` labels do not fall within the range of labels used by the nonadministrative user population on a UNICOS system.

The `syshigh` label is assigned to system-private databases, such as the user database (UDB), audit logs, and administrator-only binaries. The `syshigh` label is not dominated by any user label. This means that system files protected by the `syshigh` label cannot be read or written to by an unauthorized user.

The `syslow` label is assigned to the majority of binaries, public databases, and public directories. The `syslow` label is dominated by all user labels, but is equal to no user label. This means that system files protected by the `syslow` label can be read, but not written to by an unauthorized user.

A security administrator has the capability to use a privileged shell in order to set his or her label to the `syshigh` or `syslow` labels in order to do any necessary administrative work. See Section 8.2.4, page 167, for more information.

The MLS feature provides a default set of security labels for system files. See `/etc/privdb` for more information.

The `SECURE_MAC` configuration parameter indicates the following to system commands and daemons:

- That UNICOS file system label ranges (including `/tmp` and `/usr/tmp`) have been updated to include the `syshigh` and `syslow` labels.
- That administrative procedures have been established to adequately manage files with `syshigh` and `syslow` labels.

This parameter is set by using the `Configure system->Multilevel security (MLS) configuration->System options->Enforce system high/low security labels?` selection in the UNICOS Installation and Configuration Menu System.

Once the `SECURE_MAC` parameter has been enabled and the new configuration has been activated through the installation tool, a new kernel must be built and booted in order for the parameter to become effective.

Warning: The SECURE_MAC parameter is intended for use on UNICOS systems using the PAL-based privilege mechanism.

You can enable the SECURE_MAC parameter on systems with PRIV_SU enabled. However, administrative commands that depend on set-group-ID (setgid) functionality (instead of the root user ID) to access protected devices do not have the authority to override device label protections as required. A site can convert setgid commands to use the PAL-based privilege mechanism.

A command or daemon that needs to create syshigh or syslow labels initially has to invoke sysconf(2) to determine if it should manipulate its active label throughout its execution to create and/or manage the syshigh and syslow labels.

If the SECURE_MAC configuration option is not enabled, the calling process manages file labels as was done on UNICOS 7.0 MLS systems.

The following library routines support system labeling. Man pages for these routines can be found in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080.

Routine	Description
mls_create(3)	Allocates and creates an opaque security label for use during security label comparisons.
mls_extract(3)	Extracts a label from an opaque security structure.
mls_import(3)	Converts the text representation of the security label to the internal representation.
mls_export(3)	Converts internal security label to text representation.
mls_free(3)	Frees security label storage space.
mls_dominate(3)	Performs a security label domination test.
mls_equal(3)	Performs a security label equality test.
mls_glb(3)	Computes the greatest lower bound.
mls_lub(3)	Computes the least upper bound.

Note: Local (site) code that now performs security label comparisons should be modified to use the mls_equal and mls_dominate library routines, rather than attempting to perform their own label comparison.

8.4.2.2 File system labeling

You can use the `mkfs -L` and `mkfs -U` commands to define the minimum and maximum security levels, respectively, and the `-C` option to define the authorized compartments of a new file system. The defined values and a security label are then written to the file system, as shown in the following example:

```
/etc/mkfs -L 0 -U 5 -C 0377 /dev/dsk/usa
```

If these options are not specified, the minimum and maximum security levels of the file system default to 0 and no valid compartments are specified. This means that files assigned any other security label cannot be written to the file system. A file outside the range of the file system's security label cannot be written to the file system.

The `RC_SECLOW`, `RC_SECHIGH`, and `RC_SECMASK` configuration parameters allow an administrator to assign a label range (minimum security level, maximum security level, and compartment mask, respectively) to the `/tmp` and `/usr/tmp` file systems when the `mkfs` command is used during a reboot process. If you want to support the use of the `syshigh` and `syslow` security labels on these file systems, the `SECURE_MAC` parameter must be enabled. These parameters can be set by using the following selections in the UNICOS Installation and Configuration Menu System:

- For `RC_SECLOW`, use the `Configure system->Multilevel security (MLS) configuration->System options->/tmp and /usr/tmp minimum security level` selection
- For `RC_SECHIGH`, use the `Configure system->Multilevel security (MLS) configuration->System options->/tmp and /usr/tmp maximum security level` selection
- For `RC_SECMASK`, use the `Configure system->Multilevel security (MLS) configuration->System options->/tmp and /usr/tmp compartment mask (octal)` selection

The `-u`, `-l`, and `-c` options of the `labelit`(8) command can write an upper security level, a lower security level, and authorized compartments, respectively, to an existing mounted or unmounted file system. The `-s` option is used when installing the UNICOS MLS feature. It sets a security label on a nonsecure file system.

When used with the `-c` option, the `-s` option can change the authorized compartment set of a file after the compartments have been initially set. This includes the ability to remove as well as add compartments. When used with the `-l` or `-u` options, the `-s` option increases the minimum security level or decreases the maximum security level of a file system after these values have been initially set. You must be properly authorized to use the `-s` option for these purposes.

The following is an example of using the `labelit` command:

```
/etc/labelit -u 5 -l 0 -c 0377 /dev/dsk/usa
```

Labeling a file system with the `labelit -c` command can be done any time before it is mounted; the UNICOS MLS feature does not have to be enabled to do so.

8.4.2.3 Changing file labels

You can use the `-c`, `-l`, and `-k` options of the `spset` command to set the security compartments, security level, and flags, respectively, on files.

You must be properly authorized to use these options. This means you must have the following security attributes:

- On a system with `PRIV_SU` enabled, you must be the super user
- On a system using the PAL-based privilege mechanism, you must have an active `secadm` category.

8.4.2.4 File system access controls

On a UNICOS system, the following condition must be met to successfully mount a file system:

- If the `Configure System->Multilevel Security (MLS) Configuration->System Options->Restrict file system labels to system range?` configuration option is turned on, then the minimum and maximum security levels (with the exception of the `syslow` and/or `syshigh` security levels) and the authorized compartments of a file system must fall within the authorized ranges of the UNICOS system; otherwise the `mount` request fails.

To mount a file system on a UNICOS system with `DEV_ENFORCE_ON` set to `ON`, the device on which the file system resides must have one of the following sets of attributes:

- It must be in the `OFF` state.

- It must be in the `ON` state with the `mldev` flag on. It must also have a minimum and maximum security level range that encompasses the range of the file system and have an authorized set of security compartments that contains all the valid compartments of the file system.

Whenever the kernel is requested to assign an inode from the inode free list, the following checks are applied to the relevant file system:

- The file system is checked for a security label.
- The file system is checked to ensure that its maximum and minimum security levels and compartments bound the new file's security label.

If any of these security checks fail, the system call requiring the allocation of the inode fails, and the inode is not allocated.

8.4.2.5 File system back up operations

Enabling the UNICOS MLS feature does not alter the need for sound security practices such as file system back-up operations. All backups to tape should be stored in a secure area.

The `cpio`(1) command provide back-up operations for a file system; it processes files for the active security label of the user. On UNICOS systems using the PAL-based privilege mechanism or have the `PRIV_SU` configuration option enabled, the `cpio` command allows any user to archive data to a single-level medium. An appropriately authorized user can archive data to a multilevel medium.

Warning: Authorized users can override MAC and DAC restrictions when using the `cpio` command. Special care must be taken to ensure that data is not inadvertently downgraded.

This may occur if you do not use the `-M` and `-z` options of `cpio`. Also, when using the `-d` option, the directory is created with the label of the last object processed. If no object has been processed, the directory is created at the label of the person invoking `cpio`.

In addition, `cpio` supports the following functions:

- Allows authorized users to restore all security attributes from a multilevel medium.
- Provides the mechanism necessary to archive privilege assignment list (PAL) information in the `cpio`(5) archives.

- Minimizes the chance that security attributes in a `cpio` archive can be altered or fabricated by a non Cray ML-Safe process.
- Minimizes the risk that a multilevel `cpio` archive created by an authorized user can be read by an unauthorized user who does not have at least MAC read access to every file in the archive.

These changes to `cpio` cause the following migration issues:

- `cpio` can only restore security attributes from archives on a multilevel medium.
- For archives created with the `-z` option, redirection of `cpio` output and input must be done in a privileged shell.
- For archives created with the `-z` option, pipes connecting `cpio` output and input must be labeled as multilevel.

In addition, access control lists (ACLs) are preserved if the user is the owner of the file and has MAC write access, or is a security administrator.

The following rules must be observed when using `cpio`:

- For `cpio`, your active security label must dominate the security label of the files being copied. An appropriately authorized user can copy any file.
- To dump files, your active category must be a superset of the categories of the file.
- ACLs are preserved only if the user is the owner of the file, has an active `secadm` category, or has write access to the file.

On a UNICOS system use the `cpio -z` command to make a secure copy of a file. This option must be used when you copy in (`-i`), copy out (`-o`), or pass (`-p`) a file. In addition, the `-x` (excludes copying or preservation of ACLs), `-M` (preserving all security attributes), and `-P` (excludes copying or preservation of PAL information) options are valid only when used with the `-z` option on a system that has MLS enabled.

The `dump`(8) and `restore`(8) commands allow a security administrator to process a subset of or an entire file system with files at various security levels and compartments. These commands also dump and restore ACLs applied to files and directories.

Caution: You should take special care to ensure that dump files cannot be modified or accessed by unauthorized users.

8.4.2.6 File system security

The system or security administrator should perform the following tasks to ensure file and file system security:

- Protect memory access; there should be no public access to `/dev/kmem` (see `mem`(4)) and `/dev/mem`. Permitting public access would give users access to information not belonging to them.
- Protect access to devices; there should be no public access to the raw disk devices used for storing user files. If public access were permitted, it would be possible for a user to read from the inode list, locate the position of any information on the disk, and read it. Therefore, all entries in the `/dev` directory that pertain to disks should be owned by `root` and assigned an access mode of 600.

  On systems that have `DEV_ENFORCE_ON` enabled, raw disk devices containing file systems are protected from public access in one of the following ways:

  – By default, they are in the `OFF` state, which means only privileged processes can access them.

  – If they are in the `ON` state, they can only be mounted as file systems if they have the `mldev` flag set. This means only privileged processes can access them.

  Caution should be used to preserve these attributes when manipulating the labels of raw disk devices. For example, when changing a disk device from `OFF` to `ON`, ensure that you change it from single-level to multilevel at the same time (or before you change the state of the device).
- Protect public access to terminals. On UNICOS systems without `PRIV_SU` enabled, the default mode and access control lists (ACLs) on terminals prohibit access to the terminal by anyone but the owner. Utilities like `write`(1) override this restriction to allow interaction between users. User can use the `mesg`(1) command to allow access to their terminal through the `write` command. `write` filters its input to avoid transmission of nonprintable characters that can be interpreted as escape sequences by the recipient's terminal.

  On UNICOS non-MLS systems or systems with `PRIV_SU` enabled, this protection is not available. Users should be advised to set the mode on their terminals to 600 if they want to prevent others from writing directly to their terminals. On these systems, setting a mode of 600 disables the `write` command on that terminal and prevents direct writes to the terminal.

- Protect access to source code; generally, UNICOS source code should not be available online at customer sites. To protect UNICOS source code and system logic, the code should be assigned the `system` category.

  On systems that support the use of the `syshigh` label, this label should be used instead of the `unicos` or `system` categories to protect source code.

- Keep your file system backups in a physically secure area.
- Turn on accounting and security logging.
- Educate users on the use of UNIX file permissions, UNICOS security levels and compartments, and the `-x` option (encryption mode) of the `vi`(1), `ed`(1), or `ex`(1) commands.

In addition, a security administrator can assign one of the two following flags to a file (but not to directories) so that read or write access to that file is logged in the security log:

- `trapr`
- `trapw`

Both flags trap read and write accesses. That is, if you assign the `trapr` flag to a file, both read and write accesses to that file are logged; if you assign `trapw` to the file, the same thing happens. The functions of the flags may be split in future releases.

Use the `-k` option of the `spset`(1) command, as shown in the following example, to set these flags; you must have an active `secadm` category to set them:

```
spset -k trapr file1
spset -k trapw file2
```

8.4.2.7 File labeling

Regular files, named pipes, and sockets are assigned the active security level and active compartments of the creating subject. This information is recorded in both the memory and disk versions of the inode describing the object.

Block and character special files are created with a security label set to 0 and in the `OFF` state. The administrator must change this label to reflect the nature of the information available through the device at any given time. Use the `spdev`(8) command to label the device.

Only a properly authorized user can raise or lower the security level of a file. The security administrator can assign a security label to a file as long as the label being assigned is within the authorized range of the security administrator.

A regular file, block or character special file, or named pipe can be removed or unlinked only by a subject with the same security level and compartments as those of the object.

All nondirectory files created within a directory must have a security level and compartments equal to the security level and compartments of the directory.

See the *UNICOS Multilevel Security (MLS) Feature User's Guide*, Cray Research publication SG–2111, for more information on creating and using files on a UNICOS system.

8.4.3 Single-level and multilevel files and devices

On UNICOS systems with the MLS feature, the concept of multilevel devices has been extended to files. As a result, it is possible to have single-level and multilevel files as a general concept, and single-level and multilevel devices when rules apply to special devices.

Whether a file is single-level or multilevel depends on the nature of the data to be stored in the file. If the data in the file can be protected by a single, externally-applied label, then the file should be made a single-level file. This is the case for most files on a UNICOS system.

If the data in the file contains internal labeling information that describes the classification of specific portions of the data, the file should be made a multilevel file. An example of this type of file is the raw disk device file for a file system that contains inodes that contain labels that reflect the classifications of files.

The information stored within a single-level file is all at one security label, which is the active security label on the inode. Because the kernel knows this label, it enforces the normal MAC rules regarding access to the file and allows nonprivileged processes to gain access subject to mandatory and discretionary access controls.

The active label on a single-level file reflects the nature of the data in the file. If the file is a device special file, the data in the file is the data currently accessible through the device (for example, the data on a currently-mounted tape volume). If the file is a regular or FIFO file, the data in the file is the data that was placed there by the creator and subsequent writers to that file.

In addition to the active label, a file has a label range. This label range determines what the legal values for the active label on the file. If the file is a device special file, the label range is set by the administrator and reflects the physical security of the device, as well as any applicable site security policy issues relating to the use of the device.

If the file is a regular or FIFO file, the label range is set by the kernel and reflects the label range of the file system on which the file resides. Regardless of the type of file, the active label of the file can never be set outside the label range of the file.

On a UNICOS system, most regular and FIFO files are single-level files. An example of a single-level device special file is the tape pseudo device. This is provided by the tape daemon to a nonadministrative user when it is asked to manage tape data.

The information stored on a multilevel file is at different security labels. Each chunk of information is associated with a label embedded within the data on the file. The labels on data within a multilevel file are managed by the software that places the data in the file.

Each data object represented within a multilevel file has its own active label, so the active label of the file itself has little meaning. The label range on the file does dictates the range of differently-labeled objects that can be placed in the file. As with single-level files, the label range on a device special file is set by the administrator, while the range of a regular or FIFO file is set by the kernel, based on the file system label range.

Because the data contained in a multilevel device is at more than one label, and the labels can only be managed by the software that put them there, the normal MAC rules cannot be used to control access to multilevel files. Instead, access to multilevel files is only granted to privileged processes. Enforcement of this restriction is controlled by the `DEV_ENFORCE_ON` configuration parameter, which is explained later in this section.

An example of a multilevel file is a dump archive. It contains an archived file system, which can be shipped to a disk or across a network. An example of a multilevel device is a file system.

Note: Devices that are used as multilevel devices, but are physically secure as the system itself, can be left in the `OFF` state to signify that they can handle data from privileged processes at any label allowed by the system. This is a convenient way to handle disk devices that are well-secured and will contain file systems. The rules for access to devices in the `OFF` state are the same as for access to multilevel devices. Files cannot be placed in the `OFF` state, so they must be labeled as multilevel if they are used for multilevel data.

To make a file multilevel, you must enable the multilevel security flag (`mldev`), as shown in the following example:

```
spdev -m filename
```

In the previous example, the `-m` option of the `spdev`(8) command sets the `mldev` and `secdv` flag (the `spdev` and `state` flags are explained in the next section) and preserves all other flags with their current values. To enable the device (that is, set the `state` flag), use the `-s` option as shown in the following example:

```
spdev -m -s filename
```

The following example shows how to use the `-k` option of the `spset`(1) command to set the `mldev` flag and preserve the current value of the `secdv`. The `spset` command cannot be used to set the `secdv` flag.

```
spset -k mlsdev,secdv filename
```

The following example shows how to set or preserve the `state` flag (which is described in the next section) on a device file:

```
spdev -k mldev,state,secdv filename
```

Once a file is made multilevel, you can make it single-level by using the `spset` command to specify every flag that is currently set, except for the `mldev` flag, as shown in the following example. This example makes the file single-level, but preserves the value of the `secdv` flag:

```
spset -k secdv filename
```

The `-k` option of the `spset` command can be used to set or preserve flags (for example, the trap flags). See the `spset` man page in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022, for more information.

8.4.3.1 Assignment and access rules for labeling information

The previous section addressed only the multilevel and single-level flag (`mldev`), the active label, and the label range of files. Two other flags also control access and labeling of devices only. They are the secure device flag (`secdv`) and the state flag (`state`). The following paragraphs describe the rules for assigning labels, label ranges, and flags to devices and files:

- If the `secdv` flag is 0, the `state` flag must be off (that is `secdv` must be on in order to enable `state`).
- If `secdv` is 0, the label is invalid.
- If the `state` flag is off on a device, access is restricted to privileged processes only.
- If `mldev` is set to 1, the file is multilevel. Only a privileged process can access it. If `mldev` is set to 0, the file is single-level. It is accessible to users subject to normal MAC and DAC rules.
- The minimum and maximum label ranges of nondevice inodes are always equal to the label range of the file system on which they reside.
- The active label on a single-level inode are always within the label range of the inode.
- When mounting a file system, if the device on which the file system resides is enabled (that is, `state` is 1), the device must be multilevel (`mldev` is 1), and must have a label range that encompasses the label range of the file system.

The `setdevs`(2) system call is allowed to set the `secdv` flag to 1 or 0. The `setfflg`(2) system call is only allowed to set it to 0.

If `DEV_ENFORCE_ON` is enabled, system calls that check for MAC write access to file inodes fail if the user is not authorized and the inode is in one of the following states:

- It is a device inode and the `state` flag is not on.
- The file is multilevel (`mldev` is set to 1).

To set the `DEV_ENFORCE_ON` parameter use the `Configure system -> ->Multilevel security (MLS) configuration->System options->Enforce strict device labeling rules?` selection in the UNICOS Installation and Configuration Menu System. The default setting is `OFF`; no change in the way devices are handled should occur when set to `OFF`. This parameter must be on for a Cray ML-Safe system configuration.

Warning: The `DEV_ENFORCE_ON` parameter is intended for use on UNICOS systems using the PAL-based privilege mechanism.

You can enable the `DEV_ENFORCE_ON` parameter on systems with `PRIV_SU` enabled. However, administrative commands that depend on set-group-ID (setgid) functionality (instead of the root user ID) to access protected devices do not have the authority to override device label protections as required. A site can convert setgid commands to use the PAL-based privilege mechanism.

Enabling this parameter means that devices must be labeled before they are made available to nonprivileged users and that access to the devices is subject to the restrictions described previously.

If `DEV_ENFORCE_ON` is disabled, device labeling is enforced according to the UNICOS 7.0 implementation, although the pty behavior described in Section 8.4.3.3, page 200, is available.

Physical disk devices are always left off to ensure that only privileged processes can gain access to them. Devices constructed of multiple physical devices or logical devices residing on pieces of physical devices are used as the entry points to physical devices for normal system operations.

8.4.3.2 The `spdev`(8) command

To set file and device labels, ranges, and labeling related flags, use the `spdev`(8) command. For complete information on the `spdev`(8) command, see the man page in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022. The `spdev`(8) command uses the following options:

- The `-L` and `-K` options specify the active level and compartments, respectively, on a single-level device. These options are ignored for multilevel devices, since the kernel forces the active label on a multilevel device to the maximum label.
- With the exception of the `-C` (clearing all device labeling information on the device) and the `-p` option (printing all device labeling information on the device), all operations through the `spdev`(8) command are incremental. That means an administrator can set the label range, return and set the active label, and return again to turn the device on without having to specify all the other options for each command invocation.
- Multiple file names on the command line are allowed and apply the requested operations to each in turn.

- All user-level checks for security administrator category or `root` are removed. All operations that can be requested by `spdev` are mediated by the kernel.

The UNICOS Installation and Configuration Menu System allows the setting of minimum and maximum levels, and the authorized compartments for tape devices. Use the `Minimum security level for this group:`, `Maximum security level for this group:`, and `Maximum compartments for this group:` selections in the `Configure system->Tape configuration->configure tape resource group(s)` menu in the UNICOS Installation and Configuration Menu System.

8.4.3.3 Pseudo terminals

For pseudo terminals (ptys), the following rules apply:

- The pty label is set by the first user who opens it. The kernel labels the pty (instead of `login`).
- The master and slave labels are kept in sync by the kernel.
- A pty label automatically changes when a nonprivileged process issues a `setulvl`(2) or `setucmp`(2) system call. When a privileged process issues a `setulvl` or `setucmp` system call, the pty label does not change.

The `setusrv`(2) system call sets the label and range on a pty if executed by a privileged process.

8.4.3.4 Pty device inodes

Pseudo terminal (pty) devices provide the terminal connection emulation for a UNICOS login session. Access to a pty must be carefully controlled on a Cray ML-Safe system configuration to prevent avenues of attack. The following two avenues of attack are known to exist on a Cray ML-Safe system configuration:

- The ability of a process with write access to a pty slave device special file to subvert the physical hardware or the terminal emulation software on the other end of a login connection by using escape sequences.
- The ability of a process with read or write access to a pty slave device special file different from the one currently in use as a controlling tty of a login session to force a device driver close on the pty, thereby closing the connection.

The first attack allows a user to cause another user to issue commands without knowing it. The second attack, under certain circumstances, can result in a login session remaining active on a pty that becomes available for another login as well. This can result in unauthenticated access to the system through the previous, still active, login session.

A Cray ML-Safe system configuration ensures proper protection of pty slave device special files in the `dev` directory of the running `root` file system. However, it cannot protect access to the pty device through device special files outside the `/dev` directory of the running `root` file system.

Warning: The following information must be observed for a Cray ML-Safe system configuration.

To ensure proper protection of pty devices, do the following:

- Administrators must not create pty slave device special files outside the `/dev` directory of a `root` file system.
- When an alternative `root` file system is mounted, mount it beneath a directory that prevents search access by nonadministrative personnel. A suggested way to do this is to have all the mount points for alternative `root` file systems in a single directory that is owned by an administrative user, has an administrative owning group, and has a mode that allows search access only to the owner and the owning group. An access control list (ACL) or a `syshigh` security label can also be used to enforce this restriction.

8.4.4 `cron`(8), `batch`(1), and `at`(1) operations

For the `cron`(8) command to work with multiple labels on UNICOS systems, the `/usr/spool/crontabs` and `/usr/spool/atjobs` directories should be converted to multilevel directories (MLDs). In order for `cron` to process nonzero labeled requests, the `cron` daemon directories must also be converted to MLDs.

This allows users to have `crontab` files and one-time batch jobs at more than one label and run successfully. The label at which the job runs is the active label at the time the job was submitted with the `at`(1) `batch`(1), or `crontab`(1) commands.

If the label at which the job runs is not valid (that is, the legal range in the user database (UDB) and the label at which the job runs is no longer valid, the job at the invalid label is deleted. The `mail`(1) command attempts to send this deletion notification. Mail is received at the new security label if that label dominates the older, invalid security label.

If MLDs are used, users of `at`(1) `batch`(1), and `crontab` can use the full functionality of these commands on a UNICOS system. See ''Section 8.4.1.2.2, page 177, for more information on converting to or creating a MLD.

8.4.5 Multilevel mail operations

The `mail`(1) and `mailx`(1) commands support a multilevel mail capability. This capability is needed so that users can communicate through electronic mail without violating the UNICOS security policy.

In general, mail is delivered only to a user if the label of the sender is dominated by the maximum label of the recipient. The exception is when mail is sent with a `syshigh` label. Mail with this label is usually the output of administrative batch jobs or information sent to administrators by system daemons. It must be delivered, but it is never dominated by the maximum label of the recipient, as no user can log in with a label of `syshigh`.

Mail delivered at `syshigh` is delivered to the recipient on the local machine (unless it is addressed off of the machine by the sender. In this case, attempts are made to deliver it to the target machine. These attempts may not succeed; this depends on your network configuration).

The `.forward` file of the recipient is not processed when mail with a `syshigh` label is delivered, but the `/usr/lib/aliases` file is processed. When configuring the `/usr/lib/aliases` file, ensure that actions taken when delivering mail run safely at `syshigh` regardless of what user is receiving the mail with a `syshigh` label. Programs that permit access to or change data not directly relating to the delivery of mail should not be used in the `/usr/lib/aliases` file, nor should user-supplied programs or programs subject to change by users be allowed.

Warning: Programs within the `/usr/lib/aliases` file that are not part of the set of Cray ML-Safe components should not be used on a Cray ML-Safe system configuration. Even when using commands in the set of Cray ML-Safe components, avoid commands and command line arguments that can compromise the security policies if executed on behalf of a nontrusted recipient at the `syshigh` label.

See the *TCP/IP Network User's Guide*, Cray Research publication SG–2009, for more information on the user interface changes for trusted mail.

In addition, the following administrative changes should be made to ensure the proper execution of multilevel mail:

- Your site may want to restrict the mail files from which a user can get mail-received announcements. To do this, remove the `PRIV_MAC_READ` privilege from the privilege assignment lists (PALs) for `mail` and `mailx`.
- To allow users to save mail messages at different security labels, the user's directory that contains the saved mail message file should be converted to or created as a multilevel directory (MLD).
- The directories containing the system mail box (`/usr/mail`) and the directory used to spool queued outgoing mail (`/usr/spool/mqueue`) should be converted to or created as a MLD.

See Section 8.4.1.2.2, page 177, for more information on converting to or creating a MLD.

8.4.6 The `/proc` file system operations

When using the `/proc` file system on a UNICOS system, the following rules are observed:

- Mediation changes make it possible for a different `errno` to be returned. The difference occurs if there are multiple errors. For example, if discretionary access controls (DAC) are checked before mandatory access controls (MAC), and a process has neither DAC or MAC access, a MAC violation error is returned instead of a DAC violation error as before.
- Processes can only display files on the `/proc` file system that their label dominates.
- Cray ML-Safe processes can override MAC write restrictions.

8.4.7 `syslogd` operations

The `FORCED_SOCKET` configuration parameter controls the use of the `/dev/log` pipe. When enabled, the `syslogd`(8) command cannot use the world-writable `/dev/log` pipe. Instead, `syslogd` is forced to use the socket interface.

To enable this parameter, use the `Configure system->Multilevel security (MLS) configuration->System options->Enforce socket usage for syslogd?` selection in the UNICOS Installation and Configuration Menu System.

By default, the `FORCED_SOCKET` configuration parameter is disabled on a UNICOS 10.0 system, which means that `syslogd`(8) and `syslog`(3) continue to work as they did on UNICOS 7.0 MLS systems. If your site chooses to run a Cray ML-Safe system configuration, this configuration parameter must be enabled.

8.4.8 Destructive reads on named pipes

In previous releases of the UNICOS system with the MLS feature enabled, the mandatory access control (MAC) read policy has been enforced for reading all objects, including pipes. Because reading a pipe is destructive, this read operation is also considered a write operation. Therefore, pipes can be used by two cooperating processes to subvert the MAC policy. This is a covert channel.

The `SECURE_PIPE` configuration parameter can be used to close this covert channel. When enabled, MAC write access is required to read a pipe. When disabled, only MAC read access is needed to read a pipe.

To enable this parameter, use the `Configure system->Multilevel security (MLS) configuration->System options->Enforce restricted pipes?` selection in the UNICOS Installation and Configuration Menu System.

8.4.9 IPC objects

The System V IPC mechanism uses three named object types on the UNICOS system: shared memory segments, semaphores, and message queues. These objects have associated security label and ACL information that users need to set and get by using the `spget`(1), `spset`(1), and `spclr`(1) commands. See the associated man pages for more information on how to set and get information on these object types. See the `ipcs`(1) man page for more information on IPC objects.

In addition, IPC object creation and use can be audited on a UNICOS system. See Section 8.8.7.6, page 280, and Section 8.8.7.8, page 289, for more information.

8.5 MLS identification and authentication (I&A)

This section provides an overview of I&A security implementation and describes login and password features used on a UNICOS system. The following sections refer to `login`(1), but the information applies to the `ftpd`(8) and `rexecd`(8) daemons also.

8.5.1 Overview of I&A security implementation

A UNICOS system supports the following interactive logins:

- Through `telnet`(1B) or `rlogin`(1B) for ordinary interactive login sessions
- Through `rsh` (see `remsh`(1B)) and `rexecd`(8) for single command executions
- Through `ftp`(1B) for interactive file transfer
- Through `su`(1) for changing user identity during a session
- Through `/dev/console` through the operator workstation (OWS)
- Through `dgdemon`(8) for hardware maintenance access

The following example shown in gives an overview of the I&A sequence on a UNICOS system and highlights the security mechanisms outlined in using the `telnet` command.

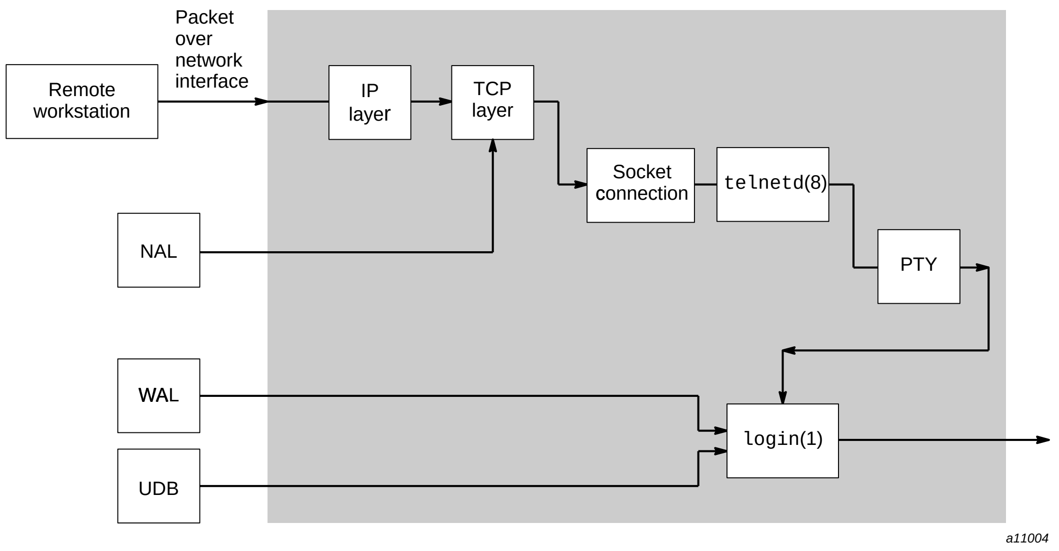

Figure 9. I&A security implementation

As shown in Figure 9, the user executes the `telnet`(1B) command to initiate the identification and authentication sequence by sending a packet and associated security label through the network interface to the IP layer.

A major function of the IP layer is to define the basic unit of data transfer on TCP/IP. The IP layer defines the security structure of these units (called an IP datagram). The security structure contains the following information:

Field	Description
Flag	Indicates a loopback packet; it is used for debugging and performance analysis. It is not sent over thenetwork.
Security Option	The IP security option: Basic Security Option (BSO) or Common IP Security Option (CIPSO).
Authority/domain	The protection authority (for BSO) or domain of interpretation (CIPSO).
Active, minimum, and maximum security labels	Active security label of the packet and the minimum and maximum label as determined by the network access list (NAL) and the network interface label ranges.

UNICOS systems support the use of the BSO and CIPSO security options. The BSO security option supports the use of security levels only; no compartments are used. The CIPSO security option supports both levels and compartments. For more information on these security options, see the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304.

The IP layer performs label format translation between the BSO/CIPSO and UNICOS representation of llabels.

The TCP layer enforces the security policy at the kernel/subject boundary for TCP users and at the system/network boundary for incoming datagrams. After receiving the IP packet from the IP layer, TCP validates an incoming datagram's active security label against the security label range of the network interface, the security label range defined in the NAL for the remote host, and the security label of the socket. A failure results in the packet being dropped and a violation logged.

The NAL controls remote host access. It grants or denies access to the local UNICOS system based on the security labels of the remote hosts (or networks). The NAL can have an entry for each remote host or network that is allowed to connect to the UNICOS system. A default entry can be used in the NAL.

Each NAL entry describes the security attributes associated with the remote host/network and the IP security option to be applied for that host's/network's communication with the local UNICOS applications. A remote host for which there is no specific host record, no applicable network entry, and there is no default NAL defined, is denied access to the UNICOS system.

After the checks are successfully completed, TCP sets the security label of the newly created socket. The label of the socket is set to the label of the incoming packet, and the label range of the socket is set to the intersection of the label range of the remote host's/network's NAL entry and the label range of the network interface.

TCP port numbers less than 1024 are considered privileged. A TCP peer that communicates using a privileged port from a Cray ML-Safe host is considered to be Cray ML-Safe. The following UNICOS processes use the privilege necessary to use these ports: `ftpd`(8), `inetd`(8), `lpd`(8), `rexecd`(8), `rlogind`(8), `rshd`(8), `telnetd`(8), `portmap`(8), and NQS.

`telnetd` opens the first pseudo pty available and sets the security label on the master and corresponding slave and makes it available for the user process.

The session is initiated with the security attributes that are determined by the centralized I&A mechanism.

8.5.2 Login procedures

The following sections describe login procedures and features on a UNICOS system. Also, the user exits for the UNICOS MLS identification and authentication mechanism are explained.

8.5.2.1 Interactive logins

On a UNICOS system, the security administrator creates and maintains the user database (UDB) that contains the following security-relevant information for each user who is allowed to access the system:

- Minimum and maximum security levels
- A default security level
- Active compartment(s)
- Authorized compartment(s)
- Minimum compartment set
- Permissions
- Maximum integrity class (obsolete)

- Active integrity class (obsolete)
- Authorized category (or categories)
- Active category (or categories)
- An encrypted password

This information is assembled for use by `login`(1) to authenticate each user who tries to access the UNICOS system.

The interactive login procedures for both a UNICOS and a Cray ML-Safe system configuration are explained in more detail in the *UNICOS Multilevel Security (MLS) Feature User's Guide*, Cray Research publication SG–2111.

8.5.2.2 Remote logins with SecurID card

The UNICOS system supports the use of the SecurID card, which is manufactured by Security Dynamics, Inc. This authentication mechanism makes it harder to break into accounts because new passcodes are generated for each authentication and a passcode cannot be used more than one time.

Note: Use of SecurID is optional on a Cray ML-Safe system configuration.

The SecurID hardware, software, and documentation must be ordered from Security Dynamics, Inc. (see "Other publications" in the preface for the address). For more information on how to use the SecurID card, refer to instructions provided by Security Dynamics, Inc.

8.5.3 Centralized identification and authentication (I&A)

The UNICOS system supports the centralized identification and authentication (I&A) mechanism.

The following commands and daemons use centralized I&A on the UNICOS system:

- `dgdemon`(8)
- `ftpd`(8)
- `login`(1)
- NQS
- `rexecd`(8)

- `rshd`(8)
- `su`(1)
- `cron`(8)

`rlogind`(8) and `telnetd`(8) indirectly use centralized I&A, as both of these daemons use `login` through the `exec`(2)) system call. `cron` uses only the `ia_mlsuser`(3) library routine.

8.5.3.1 Checks and operations

The I&A mechanism includes the following checks and operations, not all of which may be performed (depending on the request). For example, requesting passwords for batch jobs is not always necessary.

- Identify the user
- Check the password authentication
- Check SecurID authentication (if used)
- Check the workstation access list (WAL)
- Check password expiration
- Check the user's disabled flag
- Determine the session's security attributes
- Audit the successful creation of a session
- Audit the failed creation of a session and list the reason
- Update the user database (UDB) with successful or failed login information
- Disable the account after too many failed login attempts

The workstation access list (WAL) controls user access to a UNICOS system from remote hosts and workstations. It grants or denies access to services on the local system based on the identification of the user and the host or workstation from which the service request originated. The WAL can have an entry for each network address (host or workstation) that can connect to the system. If there is not an entry for the network address, it is automatically allowed to connect to the system.

Each WAL entry specifies the users and groups allowed access to a UNICOS system from that address and the services authorized for each of them to use at that address. See the *UNICOS/mk Networking Facilities Administration*, Cray Research publication SG–2604, for a complete list of supported services and the rules for gaining access through the WAL.

8.5.3.2 Library routines supporting I&A

The following library routines implement the I&A mechanism and provide a consistent, common mechanism for user identification, user authentication, auditing, user database (UDB) updating, and calculating a user's session security attributes. For more information on these routine, see the man pages in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080:

Routine	Description
`ia_user`(3)	The common UNICOS I&A authentication mechanism.
`ia_failure`(3)	The I&A failure processing routine. It manages updating the UDB entry authentication failure information, performs I&A failure auditing, and processes login delay.
`ia_success`(3)	The I&A success processing routine. It manages updating the UDB entry authentication success information and performs the I&A success auditing.
`ia_mlsuser`(3)	Determines a user's session MLS attributes.

The `ia_mlsuser` routine uses six security labels when determining the security label of a session. Two of these labels are related to the network connection, three are related to the UDB entry of the user, and the last label is a requested label. In this description, a security label is defined as x:y, where x is the security level and y is the compartment bit mask.

The network labels used are the maximum label on the connection (referred to as netmax in the following examples) and the active label on the connection (referred to as netmin in the following examples). The range of the connection is defined as netmin to netmax.

The UDB labels include the following:

- The `maxlvl` and `compart` fields, which define the maximum security label for the user. In the following examples, this label is referred to as udbmax.

- The `deflvl` and `defcomps` fields, which define the default security label for the user. In the following examples, this label is referred to as udbdef.
- The `minlvl` and `mincomps` fields, which define the minimum security label for the user. In the following examples, this label is referred to as udbmin.

The value of the `deflbl_as_minlbl` field (product `login`) of the configuration file option forces the user's default UDB security label to serve as the user's minimum UDB security label.

The range of the user's security label is defined as udbmin to udbmax. The range of the session's security label is determined first. The session's security label is defined as the intersection of the connection's security label range and the user's security label range. The session's maximum security label is the greatest lower bound (GLB) of the user's and connection's maximum security labels. The session's minimum security label is the least upper bound (LUB) of the user's and connection's minimum security labels.

LUB is the greater of the two levels and the union of the two compartment sets. The LUB of 0:011 and 1:001 is 1:011. The LUB can be thought of as the least label that dominates both labels.

The GLB is the lesser of the two levels and the intersection of the two compartment sets. The GLB of 0:011 and 1:001 is 0:001. The GLB can be thought of as the greatest label that is dominated by both labels.

The session's security label range determines the minimum and maximum security labels for the session. The user is not allowed on the system if there is no intersection between the ranges.

Next, the active security label of the session is determined. If a label is requested, the active label is set to the requested label. Access is denied if the requested label is not within the session's security label range.

If a label is not requested, the session's active label is set to udbdef, if udbdef is within the session's security label range. The session's active label is set to udbmin, if udbdef is not within the session's security label range.

The following examples show how the session's security label is determined, based on the security labels of the connection and the UDB entry values.

In the following example, assume the following values have been defined for the connection's security label range and the user's UDB entries:

```
netmin = 1:00001
netmax = 4:00001
```

```
udbmin = 0:00000
udbdef = 0:00010
udbmax = 5:00010
```

The connection's security label range is 1:00001 through 4:00001, while the user's security label range is 0:00000 through 5:00010.

Access would be denied, because there is no intersection between the connection's and user's security label ranges. The network connection has a minimum compartment set of 00001 and the user's security label range does not include this compartment set.

In the following example, assume following values have been defined for the connection's security label range and the user's UDB entries and that the `deflbl_as_minlbl` field has not be configured:

```
netmin = 1:00000
netmax = 4:01110
udbmin = 0:00000
udbdef = 2:00010
udbmax = 5:01010
```

The connection's security label range is 1:00000 through 4:01110, while the user's security label range is 0:00000 through 5:01010. The session's security label range is 1:00000 through 4:01010.

The udbdef is within the session's security label range, so the active security label is set to the value of udbdef (2:00010).

In the following example, assume following values have been defined for the connection's security label range and the user's UDB entries and that the `deflbl_as_minlbl` field has not be configured:

```
netmin = 1:00010
netmax = 1:01110
udbmin = 0:00000
udbdef = 2:00010
udbmax = 5:01010
```

The connection's security label range is 1:00010 through 1:01110, while the user's security label range is 0:00000 through 5:01010. The session's security label range is 1:00010 through 1:01010.

The udbdef is not within the session's security label range, so the active security label is set to the session's minimum label (1:00010). Access would have been denied if `deflbl_as_minlbl` was configured. The user's security label range would have had a minimum level of 2 and the maximum connection level is 1.

8.5.4 I&A user exits

Warning: The centralized I&A user exits should not be used on a Cray ML-Safe system configuration.

Seven user exits are supported on a UNICOS system. The user exits allow a site to control user I&A, including I&A success and failure process. Some examples are as follows:

- Supporting use of a local password format
- Allowing validation information to be held on a remote (that is, front-end) host
- Disallowing multiple logins
- Bypassing password processing
- Limiting access to a selected group of users or selected network address
- Disabling the use of the `su`(1) command to become `root`

The `ia_user`(3) routine supports three user exits: one at the beginning, which allows for complete replacement of the routine; a second one, which comes after normal I&A for additional site-specific authentication; and a third at the end of the routine. These routines are as follows:

User exit	Description
`ia_uex_authrep`	Provides the ability to replace functionality performed by `ia_user` or modify the `ia_user` parameters. The `ia_uex_authrep` routine has write access to all of the `ia_user` parameters, allowing `ia_uex_authrep` to modify the parameters. In addition, this routine can request that `ia_user` does not perform normal processing, but returns on return from `ia_uex_authrep`.
`ia_uex_authadd`	Provides the ability for additional authentication after successful normal authentication. It is called

	by `ia_user` after identification and the requested authentication has been performed. It is called before the password aging, maximum logins, and other login/password checking mechanisms are processed. This exit has write access to all of the `ia_user` parameters, allowing `ia_uex_authadd` to modify these parameters.
`ia_uex_authend`	Provides capability for a user exit to be called at the end of the `ia_user` routine regardless of the status that would be returned by `ia_user`. `ia_uex_authend` has write access to all of the `ia_user` parameters, allowing `ia_uex_authend` to modify the parameters.

The `ia_failure`(3) routine supports two user exits: one at the beginning, which allows for complete replacement of the routine; and a second one, which comes after normal auditing for additional site-specific auditing. These routines are as follows:

User exit	Description
`ia_uex_failure`	Provides for the complete replacement of `ia_failure` or the ability to modify the `ia_failure` parameters. The `ia_uex_failure` routine is called at the beginning of `ia_failure`.
`ia_uex_failaudit`	Provides for additional processing to be performed after normal I&A failure logging is complete, but before log-delay processing. The `ia_uex_failaudit` routine is called at the end of `ia_failure` before log-delay processing. The `ia_uex_failaudit` routine has write access to all of the `ia_failure` parameters, allowing `ia_uex_failaudit` to modify the parameters.

The `ia_success`(3) routine supports two user exits: one at the beginning, which allows for complete replacement of the routine; and a second one, which comes after normal auditing for additional site-specific auditing. These routines are as follows:

User exit	Description
`ia_uex_success`	Provides for the complete replacement of `ia_success` or the ability to modify the

	`ia_success` parameters. The `ia_uex_success` routine is called at the beginning of `ia_success`.
`ia_uex_succaudit`	Provides for additional processing to be performed after normal I&A success logging is complete. The `ia_uex_succaudit` routine is called at the end of `ia_success`. The `ia_uex_succaudit` routine has write access to all of the `ia_success` parameters, allowing `ia_uex_succaudit` to modify the parameters.

In addition, `login`(1) and `su`(1) supports the use of a user exit that is called before setting process attributes for the user's process. This routine is as follows:

User exit	Description
`ia_uex_preattr`	Provides for additional processing before the `login` and `su` processes set their attributes to that of the user. The `ia_uex_preattr` routine is called while the process is still running with special attributes (that is, as super user or with privilege).

The `ia_mlsuser`(3) routine does not support the use of user exits.

The user exits called at the beginning of the `ia_user`, `ia_failure`, and `ia_success` routines support the ability to request normal processing after returning from the user exit. If normal processing is not requested, the entire routine is bypassed after the user exit processing is completed. In this case, the corresponding user exits in the `ia_user`, `ia_failure`, and `ia_success` routines are also bypassed.

8.5.5 Password security

Most systems use passwords to regulate access to the system and prevent unauthorized individuals from logging into the system. It is important to establish a set of password guidelines for system users, educate the users on the importance of selecting, protecting, and using their passwords, and then enforce the guidelines by monitoring and auditing password use.

On a UNICOS system using a Cray ML-Safe configuration, it is of critical importance to enforce the correct use of passwords, and to use the monitoring and auditing features provided, especially in the case where repeated invalid password attempts are made.

The following list provides some rules to enforce, configuration parameters that can be used for password protection, and commands that can be used to audit password use on a UNICOS system:

- Change the `root` password often and audit its use. Limit the number of people who are allowed access to the `root` password.
- Remove old login accounts or disable them.
- Use the `CONSOLE_MSG`, `DISABLE_ACCT`, `MAXLOGS`, `LOGDELAY`, `DISABLE_TIME`, and `DELAY_MULT` parameters to protect MLS logins and passwords from unauthorized use and to lock a user's account after a specified number of failed login attempts; see Section 8.5.5.11, page 222, for more information.
- Enable the machine-generated password feature by setting the `RANDOM_PASS_ON` parameter. This feature generates a pronounceable, yet hard-to-guess password for users. See Section 8.5.5.10, page 219, for more information.
- Force users to change their passwords often by using the password aging feature; see Section 8.5.5.3, page 217, for more information.
- Use the password locking feature to lock the passwords of users on vacation or absent for any extended amount of time. See Section 8.5.5.6, page 218, for more information.
- Monitor and audit the use of passwords. This can be done in a variety of ways.
  - Use the `spcheck`(8) command to monitor failed attempts from the `su` program. See See Section 8.8.9, page 352, for more information.
  - Use the `cll`(8) command to display a user's invalid login attempts. See Section 8.5.6, page 228, for more information.
  - Use the user trap feature to monitor the activity of suspicious logins. See Section 8.5.5.7, page 218, for more information.
  - Use the `reduce` command to audit login activity from security log entries. See Section 8.8.7.9, page 292.
- Do not run `rshd`(8).

Ultimately, it is the responsibility of the security administrator to select the password features to be used, to educate system users on how to use these features, and to enforce the proposed guidelines. The remaining sections explain the password protection features in more detail.

8.5.5.1 Last login notification

At the time of each login, this feature allows the system to display the last login date, the last login time, the number of intervening login failures, and the ID of the terminal at which the user last logged in. If the user recognizes a discrepancy, the password compromise should be reported to the security administrator and investigated; see Section 8.5.6, page 228, for more information.

8.5.5.2 Generic login message

This feature allows the system to display a generic `Login incorrect` message when an unsuccessful login attempt is detected. Because it does not explicitly identify the incorrect portion of the login entry, this form of reply makes it harder to guess user names and passwords.

8.5.5.3 Password aging

Use of the `age` field in the UDB allows a maximum and minimum number to be assigned to each user password. The maximum number specifies the maximum number of weeks that the password can be used. When this limit is reached, the user is forced to change his or her password during login.

The minimum number specifies the number of weeks that a user must keep a password before changing it again. Use of the minimum number prohibits users from changing their password and then immediately changing it back to the original password. The limits assigned to each user must be greater than or equal to and less than or equal to 63. The following example shows how to use the `udbgen`(8) command to assign or change maximum or minimum numbers (the maximum number is the first number in the `age` field):

```
/etc/udbgen -c "update:jack:age:6,1:"
```

In the preceding example, Jack can use his password for a maximum of 6 weeks before being forced to change it and must use it for a minimum of 1 week before he is allowed to change it.

The system or security administrator can also force users to change their passwords before the next login attempt, as shown in the following example:

```
/etc/udbgen -c "update:jack:age:force:"
```

The `force` field forces the user to change the password during the next login attempt. See `udbgen`(8) and `udb`(5) for more information on password aging.

8.5.5.4 Password suppression

By default, this feature allows the system to suppress the display or printing of passwords supplied by users.

8.5.5.5 Password encryption

This feature allows each user password to undergo a one-way encryption before it is written to the UDB. At the time of login, the password supplied by the user is also encrypted and compared to the encrypted password in the UDB. This method prevents the display, storage, or reading of raw passwords.

8.5.5.6 Password locking

This feature allows the security or system administrator to lock a specific user password to prohibit access to the system. This may be necessary during periods of inactivity (for example, vacations). The `disabled` UDB security field is used to disable a login, as shown in the following example:

```
/etc/udbgen -c "update:jack:disabled:1
```

When 1, the user is not allowed access to the system, even if a valid password is given. A value of 0 (the default value) enables the login.

8.5.5.7 User trapping

When password compromise is suspected, this feature allows the security or system administrator to put the suspect user into a trap mode by assigning the `usrtrap` permission in the user's UDB entry.

Assigning this permission causes the system to log all auditable events, regardless of the system-wide auditing configuration (except for `SLG_STATE`). This includes all discretionary and mandatory access attempts made by the user. Use of the `usrtrap` permission results in the generation of many audit records, which means that the size of the internal kernel audit buffer may have to be increased.

To review the actions of this trapped login, use the `reduce`(8) command as shown in Section 8.5.6, page 228.

8.5.5.8 Restricted directory

The UDB contains a root directory for a user's environment; at login time, `login`(1) issues a system call to `chroot`(2), causing this directory to become the root directory for the user. This mechanism confines users to a given root directory and its subdirectories. This effectively confines the user to a subset of the file system hierarchy. Consequently, the user cannot access files in directories outside the restricted environment.

A typical restricted environment should permit access to at least the following subdirectories: `/bin`, `/etc`, `/tmp`, and `/dev`; the security administrator should place separate copies of these subdirectories in the restricted environment. This allows users in a restricted environment to execute command files residing in the restricted directory. Regardless of whether a restricted environment is established, user access to directories is always subject to mandatory and discretionary access controls.

8.5.5.9 Login attempts

This feature defines the maximum number of login attempts per connection. When this limit is reached, `login`(1) exits and the user must reestablish a connection to the UNICOS system before trying to login again. Use of this feature does not affect or supersede the use of the `MAXLOGS` or `DISABLE_TIME` parameters (see Section 8.5.5.11, page 222). Also, use of this feature does not depend on the MLS feature being enabled.

8.5.5.10 Machine-generated passwords

Although the password is usually the first line of defense in protecting access to a system, and in spite of any attempts to educate users on the importance of selecting proper passwords, many users continue to select simple, easy-to-guess passwords.

Use of machine-generated passwords enables the system to force users to select a randomly-generated password that is both easy to remember and hard for anyone else to guess, thus making password cracking harder to accomplish. You should note that use of this feature makes it impossible for users to pick their own passwords (that is, the "normal" UNICOS password mechanism is overridden when the machine-generated password feature is enabled); the only passwords available are the ones generated by this feature.

To enable the machine-generated password feature, the `RANDOM_PASS_ON` parameter must be set to `ON`. To do this, use the `Configure system->Multilevel security (MLS) configuration->Login /`

`Password options->Machine-generated passwords?` selection in the UNICOS Installation and Configuration Menu System.

In addition, the minimum length of the generated password is controlled by the `PASS_MINSIZE` parameter. The default minimum size is 8. Cray Research established this default, because use of a smaller number leads to a substantially greater chance of generating duplicate passwords.

To change the `PASS_MINSIZE` parameter, use the `Configure system->Multilevel security (MLS) configuration->Login / Password options->Minimum machine-generated password length` selection in the UNICOS Installation and Configuration Menu System.

The maximum length of the generated password is controlled by the `PASS_MAXSIZE` parameter. The default maximum size is 8. UNICOS does not support the use of passwords that are greater than 8 characters, so `PASS_MAXSIZE` cannot be set greater than 8. Also, `PASS_MAXSIZE` must be greater than or equal to `PASS_MINSIZE`; if not, the password generator forces the maximum size to be greater than or equal to the minimum size.

To change the `PASS_MAXSIZE` parameter, use the `Configure system->Multilevel security (MLS) configuration->Login / Password options->machine-generated password length` selection in the UNICOS Installation and Configuration Menu System.

The algorithm used by the machine-generated password feature generates passwords that are easy to remember, but are difficult to guess. The following list contains examples of the type of passwords generated:

- lempamdo
- coochona
- vethsymy

Note: Although the chances are extremely small, the password-generating algorithm may produce a password that may seem offensive to a user. The appearance of such a password is random and is not intended to be offensive. A user has the choice of rejecting any generated password and picking a subsequent password.

As explained in previous paragraphs, because passwords of less than 7 characters substantially increase and less than 8 increase the possibility of generating duplicate passwords, and the fact that 8 is the maximum length that can be used on a UNICOS system, a default of 8 is used for both

PASS_MINSIZE and PASS_MAXSIZE on a UNICOS system. It is recommended that your site uses the default setting of 8 for both parameters.

Enabling this feature introduces user interface changes to the passwd(1) and nu(8) command interfaces; these commands use the ranpass utility to generate the passwords. There are no changes to login(1), because it executes passwd to change passwords.

When machine-generated passwords are enabled the passwd command prompts the user with a password, as shown in the following example:

```
$ passwd
Changing password for jane
Old password:
Your new password is: kudniqui
Re-enter password or (CR) to get another:$
```

The user has the option to accept the password or to have another password generated, as shown in the following example:

```
$ passwd
Changing password for jane
Old password:
Your new password is: kudniqui
Re-enter password or (CR) to get another:
Your new password is: keltifok
Re-enter password or (CR) to get another:
Your new password is: coochona
Re-enter password or (CR) to get another:
Your new password is: rhecirou
Re-enter password or (CR) to get another:
Your new password is: osniyuib
Re-enter password or (CR) to get another:
$
```

The new password cycle continues until the user reenters the generated password, signifying that he or she accepts the password.

You cannot use the passwd -b (batch) option when machine-generated passwords are enabled.

When administrators uses the `nu -a` or `nu -m` options, they can no longer select a user's password as before, but must select one of the machine-generated passwords, as shown in the following example:

```
$ nu -a
cmd/nu/nu.c   80.4   9/25/91 15:51:03 (sn18:/etc/nu.cf60)
Login name? (1-8 characters) [quit] bob
New password is: ralcuhyd
Re-enter new password or (CR) to get another:
New password is: osniyuib
Re-enter new password or (CR) to get another:
Enter actual user name: Bob Smith
     .
     .
     .
```

After the password is selected, the `nu` sequence is the same as when the machine-generated password feature is not enabled.

When using this feature, it is important that you as a security administrator do not select a new password for yourself or other users in the presence of another person (or, at the very least, shield your screen from the other person's view). Also, when you are done selecting a password, remove or erase the screen, so that no one else can obtain the new password.

8.5.5.11 MLS login and password protection features

The UNICOS MLS feature provides six site-configurable parameters to protect and monitor the login process and the use of passwords. The parameters are as follows:

- `MAXLOGS`
- `DISABLE_ACCT`
- `DISABLE_TIME`
- `CONSOLE_MSG`
- `LOGDELAY`
- `DELAY_MULT`

All are found in `uts/cf.SN/config.h`.

Use of these parameters is site-dependent and can be used in a variety of combinations to protect MLS logins and passwords. The `CONSOLE_MSG` and `DISABLE_ACCT` parameter work independently of each other. That is, `CONSOLE_MSG` does not have to be on for `DISABLE_ACCT` to work and vice versa, although both can be on and continue to work successfully. Each of these two parameters affect how the `MAXLOGS` parameter is used; `DISABLE_ACCT` affects if `DISABLE_TIME` works.

Regardless of how the parameters are used, it is important for the security administrator to audit password usage on a daily basis and investigate any excessive or suspicious failed login attempts.

The following sections explain how to use the UNICOS installation and configuration tool to set these parameters and provide examples of how the parameters work.

8.5.5.11.1 The `MAXLOGS`, `DISABLE_ACCT`, and `DISABLE_TIME` parameters

`MAXLOGS` defines the maximum number of consecutive failed login attempts allowed to a user. For example, if `MAXLOGS` is set to 3, a user is allowed only three attempts at selecting a correct password. Whether the user is allowed to log in on the fourth (or subsequent) correct attempt depends on the state of `DISABLE_ACCT` and `DISABLE_TIME`.

The `logfails` field in a user's UDB entry is incremented each time the user makes an incorrect login attempt. When `logfails` equals or exceeds `MAXLOGS`, the account is disabled only if the `DISABLE_ACCT` parameter is `ON`. If `DISABLE_ACCT` is `OFF`, then the user's login attempts are not restricted by the limit set by `MAXLOGS`.

`DISABLE_TIME` defines the number of seconds a user is disabled from logging on after exceeding the limit set by `MAXLOGS` (assuming that `DISABLE_ACCT` is enabled). It is recommended that `DISABLE_TIME` always be set to a nonzero number. Setting this parameter to a negative number disables the user indefinitely (or until a security administrator intervenes). When using a positive number, it is recommended that you use a value no less than 60 seconds, as smaller values render this parameter ineffective.

See the examples in Section 8.5.5.11.5, page 226, for more information on the use of these parameters.

`MAXLOGS` does not affect `root` and security administrator accounts. That is, `root` can log in from the console (`/dev/console`) and authorized security administrators can log in from any terminal even though `MAXLOGS` is exceeded.

The MAXLOGS parameter is configured by using the Configure system->Multilevel security (MLS) configuration->Login / Password options->Maximum login attempts (MAXLOGS) selection in the UNICOS Installation and Configuration Menu System.

The DISABLE_ACCT parameter is configured by using the Configure system->Multilevel security (MLS) configuration->Login / Password options->Disable account after MAXLOGS attempts? selection in the UNICOS Installation and Configuration Menu System.

The DISABLE_TIME parameter is configured by using the Configure system->Multilevel security (MLS) configuration->Login / Password options->Disable time after max logins (seconds) selection in the UNICOS Installation and Configuration Menu System.

8.5.5.11.2 The CONSOLE_MSG parameter

When enabled, CONSOLE_MSG sends a message to the console when a user equals or exceeds the login limit set by MAXLOGS. See the examples in Section 8.5.5.11.4, page 225, for more information on the use of these parameters.

The CONSOLE_MSG parameter is configured by using the Configure system->Multilevel security (MLS) configuration->Login / Password options->Console messages upon reaching MAXLOGS? selection in the UNICOS Installation and Configuration Menu System.

8.5.5.11.3 The LOGDELAY and DELAY_MULT parameters

LOGDELAY defines the number of seconds that must elapse between login prompts after a failed login attempt. For example, if LOGDELAY is set to 10 seconds, then the login prompt would not appear for 10 seconds after each failed login attempt.

The LOGDELAY parameter is configured by using the Configure system->Multilevel security (MLS) configuration->Login / Password options->Seconds to delay between login tires selection in the UNICOS Installation and Configuration Menu System.

When enabled, DELAY_MULT multiplies the number of seconds defined by LOGDELAY by the number of failed login attempts to linearly lengthen the delay between each login prompt after a failed attempt. This parameter is configured by using the Configure system->Multilevel security (MLS) configuration->Login / Password options->Increment login `delay time?` selection in the UNICOS Installation and Configuration Menu System. See Section 8.5.5.11.7, page 228, for more information.

8.5.5.11.4 Using the `CONSOLE_MSG` and `MAXLOGS` parameters

When the `CONSOLE_MSG` parameter is on, and `DISABLE_ACCT` is off, `MAXLOGS` is used only to determine when a message is sent to the console. That is, when `MAXLOGS` is equaled or exceeded, a message alerts the operator or administrator that someone is attempting one or more invalid login attempts. This configuration does not prohibit a user from logging in with a correct password after `MAXLOGS` has been equaled or exceeded, however.

This situation is shown in Table 6. Assume that `CONSOLE_MSG` is `ON`, `DISABLE_ACCT` is `OFF`, `MAXLOGS` is set to 5, and `DISABLE_TIME` is set to 60.

Table 6. Login protection parameter configuration, example 1

Login sequence	User results	Console message
1 valid attempt	Login accepted	No message
1 invalid/1 valid	Login accepted	No message
4 invalid/1 valid	Login accepted	No message
5 invalid/1 valid	Login accepted	1st console message
6 invalid/1 valid	Login accepted	2nd console message

As shown in Table 6, messages are sent when `MAXLOGS` is equaled or exceeded, and the user is allowed to successfully log in when a valid password is entered. In other words, the user (malicious or otherwise) is allowed access and it is up to the operator or administrator monitoring the console to decide what to do. The table also shows that the `DISABLE_TIME` parameter is not used in this situation.

The console that receives the message is defined by the `SYSTEM_ADMIN_CONSOLE` parameter (see Section 8.7.5.2.2, page 236, for more information on this console). The appearance of the message on the console screen is preceded by a warning bell.

The format of the message is as follows:

```
Warning: user loginname has reached or exceeded MAXLOGS on
tty/pty line from host hostname
```

The fields are defined in the following list:

Field	Description
loginname	Login name of the user
tty/pty line	The device name
hostname	The name of the user's host

The following is an example of the warning message:

```
Warning: user tim has reached or exceeded MAXLOGS
on ttyp065 from host branch15
```

The warning message generated from NQS is slightly different than the one shown previously, in that the *tty/pty line* portion is replaced with `via NQS`. Otherwise, the field descriptions are the same as defined previously and NQS uses the password protection features in the same way as `login`. The following example shows a warning message for a NQS user:

```
Warning: user mary has reached or exceeded MAXLOGS via
NQS from host cray
```

8.5.5.11.5 Using the `DISABLE_ACCT`, `MAXLOGS`, and `DISABLE_TIME` parameters

When the `DISABLE_ACCT` parameter is on, and `CONSOLE_MSG` is off, `MAXLOGS` and `DISABLE_TIME` are used to prohibit incorrect login attempts.

This situation is shown in Table 7. Assume that `CONSOLE_MSG` is `OFF`, `DISABLE_ACCT` is `ON`, `MAXLOGS` is set to 5, and `DISABLE_TIME` is set to 60.

Table 7. Login protection parameters configuration, example 2

Login sequence	User results	Console message
1 valid attempt	Login accepted	Not applicable
1 invalid/1 valid	Login accepted	Not applicable
4 invalid/1 valid	Login accepted	Not applicable
5 invalid/1 valid	Login denied	Not applicable
5 invalid/ Time-out expired/1 valid	Login accepted	Not applicable

As shown in Table 7, the user is granted access until MAXLOGS is equaled (line 4 of the table). When the user makes five invalid attempts, he or she is denied access even though a valid attempt is then made. This is caused by the use of the DISABLE_TIME parameter, which defines the number of seconds a user's account is disabled after exceeding the MAXLOGS limit.

When this defined amount of time has expired, the user is allowed one more attempt at logging in. If successful, the user is granted access (as shown in line 5 of the table). If unsuccessful, then the user must again wait for the time specified by DISABLE_TIME before being allowed one more attempt to log in. Notice that no messages are sent to the console; this is because CONSOLE_MSG is off.

The Login incorrect message appears for any login attempt made after MAXLOGS is equaled or exceeded and prior to the expiration of the time limit set by DISABLE_TIME,

The DISABLE_TIME sequence continues until a successful login attempt is completed or a security administrator intervenes by resetting the login failures. As previously explained, each failed login attempt is tracked by the UDB logfails field. At no time is the logfails field set to 0 when DISABLE_TIME expires.

8.5.5.11.6 Using CONSOLE_MSG and DISABLE_ACCT parameters

When both the DISABLE_ACCT and CONSOLE_MSG parameters are set to on, then login is denied when MAXLOGS is equaled or exceeded, and messages are sent to the console. This is shown in Table 8. Assume that CONSOLE_MSG is ON, DISABLE_ACCT is ON, MAXLOGS is set to 5, and DISABLE_TIME is set to 0.

Table 8. Login protection parameters configuration, example 3

Login sequence	User results	Console message
1 valid attempt	Login accepted	No message
1 invalid/1 valid	Login accepted	No message
5 invalid/1 valid	Login denied	1st console message
5 invalid/ Time-out expired/1 valid	Login accepted	1st console message

Login sequence	User results	Console message
5 invalid/ Time-out expired/1 invalid	Login denied	2nd console message
5 invalid/ Time-out expired/2 invalid	Login denied	3rd console message

In Table 8, `DISABLE_ACCT`, `MAXLOGS`, and `DISABLE_TIME` work as explained for Table 7, page 226. In addition, because `CONSOLE_MSG` is enabled, the correct number of messages are sent to the console for invalid attempts.

8.5.5.11.7 Using the `DELAY_MULT` parameter

A site can further restrict login attempts by setting the `DELAY_MULT` parameter; when enabled, it multiplies the `LOGDELAY` parameter by the number of successive failed attempts. This lengthens the delay between each login prompt after a failed login attempt.

For example, if `LOGDELAY` is set to 10 seconds, the login prompt would not appear for 20 seconds after the second failed login attempt, 30 seconds after the third failed login attempt, and so on.

Caution: Use of the `DELAY_MULT` parameter tends to disclose information about valid users. This information becomes apparent to unauthorized users because a valid user name shows a delay between successive failed attempts.

8.5.6 Password auditing

To audit password usage, the security administrator can use the `spcheck -q` command (which reports excessive failure of the `su`(1) command) and check the output of the `reduce`(8) command, as shown in the following examples. The following example displays the users who have exceeded the limit defined by `MAXLOGS`:

```
/etc/reduce -t logn | grep Disabled
```

The following example displays a line of information that contains the user name and login ID for any user who had a password failure when logging in:

```
/etc/reduce -t logn | grep Password
```

To help determine the guesser's identity, the output of the login record through the `reduce` command identifies the host where the attempts took place. See Section 8.8.7.9, page 292, for more information on this record.

A properly authorized user can also use the `cll`(8) command to display a specific user’s failed login attempts, as shown in the following example. See Section 8.5.6.1, page 229, for a definition of a properly authorized user for the `cll` command:

```
$/etc/cll -l jack
cll: user <jack> has 0 login failures.
```

The `cll -L` command can be used to display the failed login attempts (if greater than 0) for all user logins, as shown in the following example:

```
$/etc/cll -L

User    0 <root   > has    1  login failures
User    6 <nqs    > has    5  login failures
User  128 <tom    > has   25  login failures
User  146 <alice  > has    2  login failures
User  149 <sue    > has    1  login failures
User  204 <mary   > has    7  login failures
```

8.5.6.1 Reenabling accounts

The configuration parameters explained in Section 8.5.5.11, page 222, can be used to prohibit incorrect login attempts or permanently disable a user’s account. The following information assumes that you understand how these parameters can be used to disable a user’s account.

To reenable a disabled user’s account, use the `cll`(8) or `udbgen`(8) commands, as shown in the following examples.

You must be properly authorized to use the `cll`(8) command to reset the `logfails` field in the UDB for one user or all users. Properly authorized is defined as follows:

- On a system with `PRIV_SU` enabled, you must be the super-user.
- On a system using the PAL-based privilege mechanism, you must have an active `secadm` or `sysadm` category.

The following example shows how to use the `cll -r` command to reset the `logfails` field for the user `jack`:

```
/etc/cll -r jack
```

To clear the `logfails` field in the UDB, use the `udbgen` command as shown in the following example:

```
/etc/udbgen -c "update:mary:logfails:0:"
```

You must be properly authorized to use the `udbgen`(8) command Properly authorized is defined as follows:

- On a system with `PRIV_SU` enabled, you must be the super-user.
- On a system using the PAL-based privilege mechanism, you must have an active `secadm` category to change all UDB fields. If you have an active `sysadm` category, you can change all UDB fields except for security-sensitive fields.

8.6 Object reuse

The UNICOS kernel manages shared resources (for example, process table entries, inodes, disk blocks, I/O buffers, and user memory). To ensure proper operation, these resources must be initialized when they are released to the system and reallocated. For example, when a file is removed, its data blocks are released and made available for reallocation. If the data block contents were not initialized before allocation to a different file, it would be possible for users to search for sensitive data in reallocated disk blocks.

The UNICOS kernel ensures that the user-visible contents of all the kernel-managed data resources are overwritten with zeroes or initialized to new, correct values before they are reallocated. In the case of disk blocks, the memory representation of a disk block is filled with zeroes or the data specified by the calling process whenever the block is allocated. This prevents access to the previous contents.

Most data resources are initialized at allocation time, as this saves time if the resource is never reused. Some data resources are initialized at deallocation because reuse is a certainty or there is no way to ensure correct initialization at reallocation time. In either case, the system call interface used by the UNICOS operating system prevents nonprivileged user-level processes from obtaining the contents of a data resource until it is allocated. This means there is no mechanism by which nonprivileged processes can bypass the UNICOS object reuse mechanisms.

For sites concerned about data that may remain on file system media when the media are removed, the UNICOS operating system provides the following additional mechanisms that manually or automatically overwrite disk blocks before they are released:

- The `spclr`(1) command, which overwrites the contents of files and deletes them. This allows users to ensure that the data in their files has been erased from the disk before the file is removed.
- The `SECURE_SCRUB` configuration parameter, which enables the overwriting of the contents of all disk blocks with zeroes before releasing them.

The following configuration parameters affect the behavior of the `spclr` command:

- `SANITIZE_PATTERN`
- `DECLASSIFY_DISK`
- `OVERWRITE_COUNT`
- `DECLASSIFY_PATTERN`

The `spclr -s` command overwrites disk space with a pattern determined by the `SANITIZE_PATTERN` parameter. It is recommended that the `SANITIZE_PATTERN` parameter be set to zeros. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Disk options->Scrub disk write pattern` selection in the UNICOS Installation and Configuration Menu System.

The `DECLASSIFY_PATTERN` parameter allows you to set the original overwrite pattern used when the `spclr -d` command is executed (the default pattern is 0). You can also use the `-p` option with the `-d` option. In this case, disk space is overwritten with the pattern, then with the negated pattern, and finally with a random pattern.

To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Disk options->Disk declassify write pattern` selection in the UNICOS Installation and Configuration Menu System.

The `OVERWRITE_COUNT` parameter (default is three times) determines the number of times that the `DECLASSIFY_PATTERN` is written when the `spclr -d` command is executed. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Disk options->Disk declassify overwrite count` selection in the UNICOS Installation and Configuration Menu System.

The `DECLASSIFY_DISK` parameter must be set to `YES` for the `OVERWRITE_COUNT` and `DECLASSIFY_PATTERN` parameters to work correctly.

To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Disk options->Allow disk declassification` selection in the UNICOS Installation and Configuration Menu System.

When the `SECURE_SCRUB` parameter is enabled (`ON`), the disk space is automatically overwritten once when a file is removed. It is recommended that the overwrite pattern be set to zeros (which is done by setting the `SANITIZE_PATTERN` parameter).

If the `SECURE_SCRUB` parameter is disabled (`OFF`), the disk space is not overwritten unless the user specifically executes the `spclr -s` command.

To set the `SECURE_SCRUB` parameter, use the `Configure system->Multilevel security (MLS) configuration->Disk options->Scrub data blocks on file` selection in the UNICOS Installation and Configuration Menu System. This parameter is not required by the TCSEC, because UNICOS does not allow a user to access any residual disk data.

Caution: Significant performance degradation occurs if the `SECURE_SCRUB` parameter is enabled.

8.7 MLS installation and configuration

The following sections provide MLS installation and configuration information for UNICOS and Cray ML-Safe system configurations:

- Startup and shutdown procedures
- Organization of the MLS menus used on the UNICOS Installation and Configuration Menu System (ICMS)
- Installation procedures
- Defining MLS portions of UDB entries
- Directory initialization
- Labeling commands (`privcmd`(8))
- Configuring a Cray ML-Safe system configuration

8.7.1 System startup procedure

The MLS feature, enabled by default, does not alter system startup operations. When the MLS feature is enabled, however, the following message format is displayed at the operator's console during startup:

```
SECURE_SYS.levels =  PZ_MINLVL/PZ_MAXLVL, Compartment = xxxxxxxxxxxxxxxxxxx
```

`PZ_MINLVL` (which is usually 0) and `PZ_MAXLVL` (which is usually 16) are the parameters that define the operating system's minimum and maximum security levels. The `Compartment = xxxxxxxxxxxxxxxxxxx` field defines the operating system's compartment set. The following example shows the display:

```
SECURE_SYS levels =  0/16, compartment = 0777777777777777777777
```

You can use the `spset -s` command to change the security level and compartment for the operating system. You must be properly authorized to use this option. See Section 8.7.5.2.1, page 235, for more information.

You can use the `spget -s` command to display the system's security label. Anyone can use this option to display the information.

During system startup (even after a system failure), the `/dev/tty*` entries are automatically accessed and cleared by `spwcard`(8).

8.7.2 Subsystem startup procedure

Daemons are started automatically during system startup from within the `/etc/rc` file through the `/etc/sdaemon` command, which is executed on entry to the multiuser state. You must be a properly authorized administrator to manually start or restart a daemon.

8.7.3 System shutdown procedure

The MLS feature, enabled by default, does not alter system shutdown operations. The `shutdown`(8) command sends a shutdown warning message on a UNICOS system through the `wall`(8) command to all users regardless of their security labels.

If you want to send a separate message to all users, then you must be properly authorized when executing the `wall` command. If you are not properly authorized then only those users executing at the same security label and have messages turned on receive the message.

A properly authorized user is defined as the super user on systems with `PRIV_SU` enabled; for systems using the PAL-based privilege mechanism, properly authorized is defined as having an active `secadm` or `sysadm` category.

8.7.4 System clearing procedure

Before the UNICOS system is started, the Cray System Clear package can be used to clear (or scrub) the Cray Research hardware environment.

The Cray System Clear utility is a stand-alone routine that writes over the data in all memory (mainframe, IOS buffer memory and IOP local memory), stacks, internal buffers, and registers, as well as on selected disks and specified high-speed channel buffers. The interactive session or a shell script can be used to activate and control the hardware scrubbing process.

It is important to note that this utility has not been integrated with the UNICOS startup process; therefore, system scrubbing operations must be initiated manually. See your Cray Research engineer for instructions on how to run this utility.

Note: The Cray System Clear utility is not supported for use on a Cray ML-Safe system configuration.

8.7.5 MLS configuration parameters

The following files are used to configure the MLS feature on a UNICOS system:

- `sys/secparm.h`
- `uts/cf.SN/config.h`
- `cf/seclabs.c`

The `secparm.h` file cannot be configured by your site; the remaining two files can be changed by the site. The following sections describe these files.

8.7.5.1 The `secparm.h` file

In pre-6.0 releases, the `secparm.h` file contained all the security parameters that could be configured by a site. For post-6.0 releases, it contains parameters that are used by the UNICOS kernel and cannot be changed by site personnel.

The file contains definitions for the following:

- The names and representations for the user permissions

- The definitions of the process 0 initial security parameters
- The macros used to identify certain Cray ML-Safe processes

8.7.5.2 The `uts/cf.SN/config.h` file

The `uts/cf.SN/config.h` file contains all the parameters that can be configured by the site (except for the site-specific level naming conventions and the site-specific compartment definitions, which are defined in the `seclabs.c` file). These `uts/cf.SN/config.h` parameters can be set by using the `Configure system->Multilevel security (MLS) configuration->`*appropriate selections* in the UNICOS Installation and Configuration Menu System.

The *appropriate selections* are as follows:

- `System options`
- `Network security options`
- `Login / Password options`
- `Disk options`
- `Security log configuration`
- `Import the security configuration ...`

Each security submenu has help files that explain the various selection or refers you to the proper documentation in this manual. The selections found in these menus are documented throughout the relevant sections of this MLS chapter and are not be repeated here, except for the following selections, which did not fall into any other discussion.

See the *UNICOS System Configuration Using ICMS*, Cray Research publication SG–2412, for more information on the selections and help files available for all MLS configuration menus.

8.7.5.2.1 Setting the system's security label

The operating system's default minimum and maximum security levels (`MINSLEVEL` and `MAXSLEVEL`, respectively) are defined in `/uts/cf.SN/config.h`.

To configure `MINSLEVEL`, use the `Configure system->Multilevel security (MLS) configuration->System options->Minimum` `security level` selection in the UNICOS Installation and Configuration Menu System.

To configure `MAXSLEVEL`, use the `Configure system->Multilevel security (MLS) configuration->System options->Maximum security level` selection in the UNICOS Installation and Configuration Menu System. This information is resident in memory.

The operating system's valid set of compartments (`SYSVCOMPS`) is also defined in `uts/cf.SN/config.h`. This octal mask can be set by using the `Configure system->Multilevel security (MLS) configuration->System options->Valid system compartment mask (octal)` selection in the UNICOS Installation and Configuration Menu System. For a compartment to be available on the system, its associated bit setting in this octal mask must be set to 1.

An appropriately authorized user can use the `spset -s` command to change this mask. See the *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011, for more information on the `spset`(1) command.

8.7.5.2.2 Configuring consoles

Access to the operator or administrator ID can be restricted to a console through use of the following parameter, which is found in `uts/cf.SN/config.h`:

Parameter	Description
`SYSTEM_ADMIN_CONSOLE`	Allows the site to set the system console to the named console; reset at time of system initialization. This parameter must be set to `/dev/console`.
`SECURE_SYSTEM_CONSOLE`	Use of this parameter is obsolete as of the UNICOS 9.2 release.
`SECURE_OPERATOR_CONSOLE`	Use of this parameter is obsolete as of the UNICOS 9.2 release.

To set the `SYSTEM_ADMIN_CONSOLE`, use the `Configure system->Multilevel security (MLS) configuration->System options->Default system console` selection in the UNICOS Installation and Configuration Menu System.

Note: These parameters will no longer be available in the UNICOS 10.0 release.

8.7.5.3 The `seclabs.c` file

The `seclabs.c` file contains the site-specific compartment definitions and the security level naming options. You can define these parameters by selecting `Configure system->Multilevel security (MLS) configuration->Site labels configuration->Compartments` or `Levels` selections in the UNICOS Installation and Configuration Menu System.

8.7.5.3.1 Security level naming

If the security level names were imported during the installation process, then, by default, security levels (0, 1, 2,...) are assigned names (`level0`, `level1`, and so on). These security level names can be changed. For example, `level0` could be named `public`, `level2` could be named `private`, and so on.

It is possible to define new, or redefine existing security level names and values without rebuilding commands, utilities, and libraries (only the UNICOS kernel must be rebuilt). This capability is supported by using the `getsectab`(2) system call and `sectab`(5) structure. The UNICOS system allows the `getsectab`(2) system call to return the configured security tables, regardless of whether MLS is enabled. This allows you to to configure various MLS information (for example, security-relevant fields in the user database (UDB)) before actually booting a UNICOS MLS kernel.

8.7.5.3.2 Compartment definition

Compartment use and definition should be determined by the security personnel before a Cray ML-Safe configuration of the system is installed. Redefining compartments after the system is installed and running can present many security holes.

For example, redefining a compartment, and then assigning the new compartment to a new user, could possibly grant the new user unwanted permission to an object labeled with the old compartment definition.

8.7.5.4 Permission definitions

The following paragraphs clarify the definitions and differences between the following user permissions:

- `permbits` (permission bits for user permissions)
- `permits` (permission bits for user permissions on UNICOS systems)
- `sitebits` (permission bits reserved for site definition)

These permissions are defined in the user database (UDB) on UNICOS and Cray ML-Safe system configurations. These permissions are assigned to a user at login. See the `udbgen`(8) man page for more information on assigning these permissions. The permissions are used throughout the session unless explicitly turned off by a process.

`permbits` are used on both UNICOS systems with MLS and without MLS, and provide users the authority to perform certain functions such as changing the ownership of their files. `permbits` are not used within the Cray ML-Safe configuration kernel for determining privileges, although the tape subsystem uses the `tape-manage`, `bypasslabel`, and `wrunlab` permbits.

`permits` are similar in function to `permbits`. When `PRIV_SU` is enabled, the UNICOS system uses the `suidgid` and `usrtrap` `permits`. These `permits` have no effect on systems using the PAL-based privilege mechanism. The `permits` are permanently defined in the security parameter file (`sys/secparm.h`).

The `suidgid` permission gives the user explicit permission to set the set-user-ID (setuid) and/or set-group-ID (setgid) bits for a file. Restricted management of setuid and setgid files is enforced only on UNICOS systems with the `FSETID_RESTRICT` configuration parameter enabled. This permission is used only on systems with `PRIV_SU` enabled.

The `usrtrap` permission can be used on any UNICOS configuration. This permission is not, by a true definition of the word, a permission, but a process attribute. It is described here, as it is defined in `sys/secparm.h`. The `usrtrap` permission sets the user in trap mode, causing the system to log all auditable events, regardless of the system-wide auditing configuration (except for `SLG_STATE`). This includes all discretionary and mandatory access attempts by this user.

For example, if `SLG_ALL_VALID` is disabled, but the user has the `usrtrap` permission assigned, all valid file access attempts for the user are still logged. For more information on the security log, see Section 8.8.1, page 263. Do not assign the `usrtrap` permission to the security administrator.

You may see instances where the the `reclsfy`, `lbypass`, `wrunlab`, and `install` permissions are displayed in audit records and other UNICOS outputs. These are obsolete `permits` and they have no effect on a UNICOS system.

`sitebits` are similar in function to `permbits`, but the meaning of each bit is defined by the site. Their usefulness is limited only by the site's need to maintain compatibility with other UNICOS sites. They can be used on a UNICOS system with `PRIV_SU` enabled, with the understanding that the site is aware of how any security-related `sitebits` could affect the default security policies enforced on either of these security configurations. The introduction of site-written code that uses `sitebits` to enforce security policies on a Cray ML-Safe system configuration should not be used unless proper accreditation procedures are followed.

8.7.6 Defining MLS UDB entries

The user database (UDB) has fields that pertain specifically to the UNICOS MLS feature. These fields include the following:

Field	Description
`maxlvl`	Maximum security level
`minlvl`	Minimum security level
`defcomps`	Active compartments
`deflvl`	Default security level
`mincomps`	Minimum compartment set
`comparts`	Authorized compartments
`permits`	User permissions
`intcat`	Active category
`valcat`	Authorized categories

The `udbgen`(8) command creates and maintains the UDB.

Before booting a UNICOS system to multiuser mode, you should define all users in the UDB. All user entries should be carefully defined, because a user with more privileges than necessary can (inadvertently or maliciously) corrupt or destroy system or user data.

Defining the security administrator entry is of special importance to successfully installing and maintaining the UNICOS system. Administrative user entries should be set up according to the following guidelines:

- Assign each administrative user a unique, non-root login ID. If you need more than one person to function as the same type of administrator on non-`PRIV_SU` systems, define a unique login ID for each user and assign each login account the same administrative category. On UNICOS systems that depend on the super user for administration (that is, the system has `PRIV_SU` enabled), require administrators to use the `su`(1) command to become the `root` user.

Table 9 gives guidelines for defining the UDB security fields for administrators, operators, and general users on a UNICOS system.

Table 9. Suggested values for UDB security fields

UDB	Security administrator	System administrator	System operator	Users
`minlvl`	0	0	0	0
`maxlvl`	16	*	*	*
`deflvl`	0	0	0	0
`comparts`	*	*	*	*
`defcomps`	None	None	None	None
`mincomps`	0	0	0	0
`permits`	`suidgid`	`suidgid`	`suidgid`	**
`intcat`	None	None	None	None
`valcat`	`secadm`	`sysadm`	`sysops`	None

* Definition of this field is site-specific.

**Assign the `suidgid` only if necessary. Assign the `usrtrap` permission if you with to log all the user's discretionary and mandatory access requests in the security log.

In addition to establishing a user's UDB entry, the security administrator should ensure that the user's home directory and any initial files (for example, `.profile`, `.cshrc`, and `.login`) are labeled to match the user's UDB entry.

8.7.7 Directory initialization procedures

Several directories must be either converted to multilevel directories (MLDs) or assigned the wildcard label to enable the concurrent processing of user clearances and file classifications at multiple security labels.

Assigning the wildcard label to many of the directories is done at system startup time by the `spwcard`(8) command, which is run automatically from the `/etc/rc` script. NQS and the tape daemon also assign the needed wildcard labels through their respective initialization routines. See Section 8.4.1.2.2, page 177, and Section 8.4.1.2.1, page 176, for more information on converting to MLDs and wildcard directories, respectively.

> **Note:** Wildcard directories cannot be used on a Cray ML-Safe system configuration. Where wildcard labels were used on UNICOS systems (that is, non-Cray ML-Safe UNICOS systems), they have been replaced with MLDs. If your site has already configured the system to support a Cray ML-Safe system configuration, using the `spwcard` command does not relabel any file or directory (for example, relabel a MLD with a wildcard label).
>
> The tape daemon and the NQS installation and startup procedures also do not support the use of wildcard labels on a Cray ML-Safe system configuration.

The `spwcard` command assigns a wildcard label to the following directories when it is executed by the `/etc/rc` script:

- `/tmp`
- `/usr/tmp`
- `/usr/mail`
- `/usr/spool/mqueue`

During the initial installation of NQS, the `qstart` utility executes `qconfigchk` and looks for the `NQE_NQS_MAC_DIRECTORY` variable. If the variable is set to 1, then the NQS spool directories will be created as MLDs.

If you are upgrading your NQS configuration to use MLDs, you must do the conversion manually. For more information on converting the NQS spool directories, see the *NQE Administration*, Cray Research publication SG–2150.

The directories assigned the wildcard security label and automatically set by the NQS process are as follows:

- `/usr/spool/nqs/private/root/control`

- `/usr/spool/nqs/private/root/data`
- `/usr/spool/nqs/private/root/failed`
- `/usr/spool/nqs/private/root/interproc`
- `/usr/spool/nqs/private/root/output`
- `/usr/spool/nqs/private/root/chkpnt`
- `/usr/spool/nqs/private/root/FIFO`
- `/usr/spool/nqs/private/root/LOGFIFO`
- `/usr/spool/nqs/private/root/reconnect`
- `/usr/spool/nqs/private/requests`

8.7.8 The `privcmd`(8) command

The `privcmd`(8) command sets file privileges, the security label, permissions mode, owner, owning group, and security flags on system objects. For more information on using this command, see the `privcmd` man page in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022.

8.7.9 MLS installation and configuration procedures

Warning: In UNICOS 9.2 and later releases, all sites are required to assign PALs. The supported privilege configurations are as follows:

- PALs augmented by `PRIV_SU`
- PALs only

There are several procedures you can use for configuring the MLS feature on a UNICOS 10.0 system, depending on whether you are performing an initial install or an upgrade. The following three procedures are documented in *UNICOS System Configuration Using ICMS*, Cray Research publication SG–2412:

- An initial UNICOS 10.0 installation (with MLS)
- A UNICOS 9.3 system with MLS upgraded to a UNICOS 10.0 system with MLS
- A UNICOS 9.1 or earlier system without MLS upgraded to a UNICOS 10.0 system with MLS

The procedures for migrating to a Cray ML-Safe system configuration and migrating from a single-level UNICOS system to a multilevel UNICOS system are outlined in the following sections:

- A UNICOS 10.0 system with MLS to a Cray ML-Safe system configuration
- A single-level UNICOS system to a multilevel UNICOS system

8.7.9.1 Cray ML-Safe configuration

A Cray ML-Safe configuration of the UNICOS system is established when the specified configuration of the UNICOS system is enabled. In order to install the UNICOS portion of a Cray ML-Safe configuration, the following documentation must be used, depending on your system:

- *UNICOS Installation Guide for CRAY J90se GigaRing based Systems*, Cray Research publication SG–5296
- *UNICOS Installation Guide for CRAY C90, CRAY T90, and CRAY T90 IEEE Model E based Systems*, Cray Research publication SG–5297
- *UNICOS Installation Guide for CRAY T90 and CRAY T90 IEEE GigaRing based Systems*, Cray Research publication SG–5298

In addition, several software releases and related documentation are needed for the proper installation of other Cray ML-Safe software components. Installation manuals include:

- *NQE Administration*, Cray Research publication SG–2150
- *UNICOS System Configuration Using ICMS*, Cray Research publication SG–2412
- *OWS x.x Release and Installation Notes*, Cray Research publication RN–5055
- *Support System and IOS-E Release Overview*, Cray Research publication RO–5060
- *Cray Data Migration Facility (DMF) Release and Installation Guide*, Cray Research publication SG–5166
- *SWS-ION Release Overview*, Cray Research publication RO–5292

The following procedures assume you have already installed and are running a UNICOS 10.0 system.

Before you configure a Cray ML-Safe system, you should make the following preparations:

- You must have the security administrator, system administrator, and system operator roles defined by assigning the correct category to the appropriate site personnel before installing the Cray ML-Safe configuration. For more information, see Section 8.7.6, page 239.
- You should review your `crontab` jobs before migrating to a Cray ML-Safe configuration. The following example shows the type of change that should be made; failure to make these changes means that the jobs will fail on a Cray ML-Safe configuration, as they call commands that require privilege. For example, change the following in `root crontab` as shown:

  ```
  su sys -c "/usr/lib/sa/sa1"
  ```

  to

  ```
  setucat secadm >/dev/null; su sys -c "/usr/lib/sa/sa1"
  ```

  The `umask` setting in `skl/etc/profile` and `skl/etc/cshrc` must be copied to `/etc/profile` and `/etc/cshrc`, respectively, to ensure that the proper setting of 077 is used. For more information, see Section 8.3.1, page 172
- The trivial file transfer protocol (TFTP) must be disabled on a Cray ML-Safe configuration. To disable this feature, use the `Configure system->Network Configuration->TCP/IP Configuration->Generic Internet Daemon Configuration` selection in the UNICOS Installation and Configuration Menu System.

In addition to these preparations, the following products should be installed before installing a Cray ML-Safe configuration:

- If your site is running the Cray Data Migration Facility (DMF), refer to the *Cray Data Migration Facility (DMF) Release and Installation Guide*, Cray Research publication SG–5166, for installation instructions for a Cray ML-Safe configuration.
- If your site is using online tapes for nonadministrative users, the Cray/REELlibrarian (CRL) must be installed. CRL is not required for administrative-only access to tapes. Refer to the *Cray/REELlibrarian (CRL) Administrator's Guide*, Cray Research publication SG–2127, for installation instructions for a Cray ML-Safe configuration.

- On UNICOS systems using an IOS-E, install the operator interface (`opi`) in Cray ML-Safe mode. Refer to the *OWS Operator Workstation Administrator's Guide*, publication SG-3038, for installation instructions for a Cray ML-Safe configuration. The OWS-E interface must have the `admin` option set.

Use the following instructions to build and install a Cray ML-Safe configuration. Unless otherwise noted, these steps may be completed in any order prior to booting the Cray ML-Safe kernel. Cray Research recommends that this procedure be done in dedicated time.

1. If the system is in multiuser mode, go to single-user mode and unmount all file systems. Sync the file system and flush the ldcache (if any).

2. Label the `root`, `/usr`, `/tmp`, `spool`, `src`, the file system on which the job temporary directories reside, and the file system on which the security logs reside (by default, this is `/usr/adm/sl`, although it can be site-defined) with a `syslow` to `syshigh` label range by using the `labelit`(8) command, as shown in the following example. The `core` file system must be labeled with a `syshigh` label.

```
/etc/labelit -l syslow -u syshigh /dev/dsk/usr
/etc/labelit -l syslow -u syshigh /dev/dsk/tmp
/etc/labelit -l syslow -u syshigh /dev/dsk/spool
/etc/labelit -l syslow -u syshigh /dev/dsk/src
/etc/labelit -l syslow -u syshigh /dev/dsk/jtmp
/etc/labelit -l syslow -u syshigh /dev/dsk/usr_adm_sl
/etc/labelit -u syshigh /dev/dsk/core
/etc/labelit -l syslow -u syshigh /dev/dsk/root
```

Caution: Do not sync the file systems after this point.

3. Reboot your initial UNICOS 10.0 `PRIV_SU` system, which makes the new label range on the `root` file system effective. Go to multiuser mode.

4. As `root`, prepare for entering the Installation and Configuration Menu System by ensuring the `src` file system is mounted. If it is not mounted, mount it at this time, as shown in the following example:

```
/etc/mount /usr/src
```

Set the `TERM` environment variable so you can run the Installation and Configuration Menu System, as shown in the following example:

```
export TERM=xterm
eval 'resize'
```

Run the Installation and Configuration Menu System, as shown in the following example:

```
cd /etc/install
./install
```

5. For any product that is specified as `YES` in the UNICOS Installation and Configuration Menu System, the menu system must automate the configuration. To ensure that the settings are set to `YES`, check the following product settings in the `Configure system->Configurator automation options` menu selection:

```
Major software configuration?                YES
Mainframe hardware configuration?            YES
IOS configuration?                           YES
Kernel configuration?                        YES
Multilevel security (MLS) configuration?     YES
Tape configuration?                          YES
Cray/REELlibrarian configuration?            YES
Host address configuration?                  YES
Network address configuration?               YES
Services configuration?                      YES
Network interface configuration?             YES
Network hardware address configuration?      YES
TCP/IP configuration?                        YES
TCP/IP protocols configuration?              YES
TCP/IP lookup configuration?                 YES
NFS configuration?                           YES
NQS configuration?                           YES
System daemons configuration?                YES
Startup (/etc/rc) configuration?             YES
```

6. If you have manually changed any configuration files since the last system build, or you are now automating a major configuration item from the previous step, you must import those files at this time. If the files are not imported, this information is lost. For instructions on importing UNICOS 9.3 configuration information, see *UNICOS System Configuration Using ICMS*, Cray Research publication SG–2412.

7. In the next step, you make the necessary changes for a Cray ML-Safe configuration. Before doing so, check your NAL set configuration. Any NAL set name that has security ranges lower than `B1` must be removed. Change any of the NAL sets that are needed at this time. All NAL entries must be `B1` or greater on a Cray ML-Safe configuration.

8. Enable the Cray ML-Safe configuration by manually setting the following items in the Installation and Configuration Menu System.

   In the `Configure System->Major software configuration` menu selection, make the following settings:

```
Kerberos network data encryption: OFF
Network Information Service (NIS): OFF
Cray based network monitor: OFF
Network testing tools: OFF
DCE Distributed File Service (DFS): OFF
File Transfer Agent (FTA): OFF
```

   In the `Configure System->Multilevel security (MLS) configuration` menu selection, make the following settings.

   System options:

```
Enforce system high/low security labels?  ON
  /tmp and /usr/tmp minimum security level: SYSLOW
  /tmp and /usr/tmp maximum security level: SYSHIGH
  Enforce strict device labeling rules?  ON
  Enforce socket usage for syslogd?  ON
  Super-user privilege policy?  OFF
```

   Network security options:

```
Strict B1 evaluation rules: YES
  Default to multi-level privileged sockets for compatibility: NO
  Traditional hosts.equiv & .rhosts: NO
```

   In the `Configure System->Multilevel security (MLS) configuration->Network security options->Network-Protocols Security Configuration` menu selection, make the following settings.

   MLS Network Access List (NAL) Sets:

```
Security class: Must be B1, B2, B3, or A1. D, C1, and C2 are not evaluated.
```

   MLS Network Security Definitions:

```
Item type: Must be either ip_host or ip_net. 'station' is not evaluated
```

   In the `Configure System->Tape configuration->Select tape subsystem options` menu selection, make the following settings.

General options:

```
Front end servicing at startup: NO
  Allow unprotected tapes: NO
  Secure front end: NO
  Ask operator permission to switch label type: YES
```

Check options:

```
Check file identifier: YES
  Check protection flags: YES
```

Default options:

```
Is servicing front end mandatory?  NO
  Operator verify each scratch mount: YES
```

In the `Configure System->Network configuration->General network configuration->Network interface configuration` menu selection, make the following setting:

```
Address family: inet [afnet]
```

In the `Configure System->Network configuration->TCP/IP configuration` menu selection, make the following settings:

Host/address lookup:

```
Use Domain Name (DN) service?  NO
```

Kernel parameters:

```
IP forwarding?  NO
```

In the `Configure System->Network configuration->NFS configuration->List of exported file systems` menu selection, make the following setting:

```
  Require Kerberos authentication (krb): <NULL> [cannot set to ’krb’]
```

In the `Configure System->Ctartup (/etc/rc) configuration` menu selection, make the following setting:

```
Start system accounting?  NO
```

9. Exit the current menu by pressing `e`. Activate the configuration by using the `Configure system->Activate the configuration...` selection.

10. Build the new Cray ML-Safe kernel by using the `Build/Install System` menu and setting the build selections as shown in the following example:

```
M-> Build options ==>
    /usr/src reconfiguration files ==>
    Build action to take               install
    Build object                       all objects
    Components to build                specific component
    Major components selection ==>
    Specific component to build        uts
    Do the build in batch?                        NO
    NQS submission options ==>
    Do the build ...
    Restart the build ==>
    Review last build summary ...
    Escape to a chroot shell ...
```

Execute the build by selecting the `Do the build ...` selection. This results in building and installing a Cray ML-Safe system configuration.

Note: If you build and install any product on a Cray ML-Safe system, it must be done from within the Installation and Configuration Menu System. For products that are not currently supported on the menu system, the administrator must invoke the menu system and then move to a shell before attempting to build and install the product. This is necessary, because the base Installation and Configuration Menu System binary enables the necessary privileges to build and install the various UNICOS binaries. The following example shows the sequence:

```
# setucat secadm
# cd /etc/install
# ./install
```

Then, from the Installation and Configuration Menu System, you would escape to a `chroot` shell as follows:

```
M-> Build/Install System ==>
       A-> Escape to a chroot shell ...
```

11. Transfer the Cray ML-Safe kernel to the workstation (SWS, OWS, or console) by hand. For lists of files and their locations, see *UNICOS System Configuration Using ICMS*, Cray Research publication SG–2412. Keep the old `PRIV_SU` kernel on the workstation. If the Cray ML-Safe kernel will not boot into multiuser mode, the old `PRIV_SU` kernel can be booted into single-user mode to aid in analyzing the problem. Events such as incorrect

kernel configuration or incorrect or missing system file labels and PALs can prevent the booting of a Cray ML-Safe kernel.

12. Escape to the shell by using the `!` command.

13. Configure NQS by using the NQE configuration tool instead of the Installation and Configuration Menu System. Configurable MLS features are disabled by default. For information on configuring NQS, see *NQE Administration*, Cray Research publication SG–2150.

    The following list summarizes the configuration changes for NQS on a Cray ML-Safe configuration:

    - Set the following parameters in the `/etc/nqeinfo` configuration file:

      ```
      NQE_NQS_MAC_COMMAND        1
      NQE_NQS_MAC_DIRECTORY      1
      NQE_NQS_PRIV_FIFO          1
      ```

    - Ensure that NQS user validation is set to either `password` or `file` (`no_validation` is not acceptable for a Cray ML-Safe configuration). Edit the NQS configuration file and search for the `set validation` configuration command. The argument must be either `password` or `file`. If the argument is `no_validation`, change it to `password` or `file`.

    - Ensure that the NQS configuration does not use FTA as an output agent. Edit the NQS configuration file text and search for `output_agent`. For each machine ID that currently has FTA as an output agent, delete the FTA line.

    - Ensure that all the NQS spool directories, logfile, and console file are on a file system with a `syslow` to `syshigh` label range.

      The Installation and Configuration Menu System should be set as follows:

      ```
      Configure system ==>
        Multilevel security (MLS) configuration ==>
          System Options ==>
            Enforce system high/low security labels ==>  ON
      ```

    - Convert all NQS wildcard directories to MLDs. The `nqsdaemon` must not be running when you convert the directories. For information on converting directories, see *NQE Administration*, Cray Research publication SG–2150.

The following is a list of directories that must be converted (`NQE_NQS_SPOOL` is defined in `/etc/nqeinfo`):

```
NQE_NQS_SPOOL/private/requests
NQE_NQS_SPOOL/private/root/chkpnt
NQE_NQS_SPOOL/private/root/control
NQE_NQS_SPOOL/private/root/data
NQS_SPOOL/private/root/failed
NQE_NQS_SPOOL/private/root/interproc
NQE_NQS_SPOOL/private/root/output
NQE_NQS_SPOOL/private/root/reconnect
```

14. If you have not already done so, all wildcard labeled directories must be converted to MLDs. The following list contains the directories that require the use of MLDs on a Cray ML-Safe configuration:

    - `/usr/mail`
    - `/usr/spool/mqueue`
    - `/usr/spool/cron/crontabs`
    - `/usr/spool/cron/atjobs`
    - `lpr`(1) and `lpd`(8) spool directories (`/usr/spool/*`)
    - `/tmp`
    - `/usr/tmp`

If you want to preserve existing job temporary directories and carry them forward to the new system, then each existing job temporary directory must be converted into a MLD under the `/tmp.mld/jtmp` directory.

Note: The conversion of `/tmp` and `/usr/tmp` must be done in single-user mode, as some of the daemons use this directory while in multiuser mode.

See Section 8.4.1.2.4, page 181, for more information on converting the directories.

15. The `umask` setting as shown in `skl/c1/etc/profile` and `skl/c1/etc/cshrc` must be copied to `/etc/profile` and `/etc/cshrc`, respectively. The `umask` setting for a Cray ML-Safe configuration must be 077.

16. The device labels on the various tty lines to the OWS are assigned by `getty`(8) via the `/etc/inittab` file. The file must be set as shown in the following example. Any other tty line that is in the `/etc/inittab` file should be changed with the `getty -L -C` command, as shown in the following example:

    ```
    co::respawn:/etc/getty -L 0-16 -C 0-07777777777777777777777
    console console
    01:2:respawn:/etc/getty -L 0-16 -C 0-07777777777777777777777
    tty01 9600
    02:2:respawn:/etc/getty -L 0-16 -C 0-07777777777777777777777
    tty02 9600
    ```

    Cray platforms support several lines to the OWS. Access to these devices is controlled through `opi` in the Cray ML-Safe mode from the OWS side and by the device labels on the UNICOS side.

17. Change the label range on the local host interface from `syslow` to `syshigh` and 0 compartments to all available compartments. Go to the `Configure system->Network configuration->General Network Configuration->Network interface configuration selection` and select the `localhost (lo0)`. Change the following items: Minimum security label to `syslow`; Maximum security label to `syshigh`; Maximum security compartments to `07777777777777777777777`.

18. Ensure that the `root`, `/usr`, `/src`, and administrative file systems have been mounted, as follows:

    ```
    /etc/mount /usr
    /etc/mount /usr/src
    /etc/mount /usr/spool  # if site has one
    /etc/mount /usr/adm    # if site has one
    /etc/mount /usr/adm/sl # if site has one
    ```

19. Two objects are not included in the base Cray ML-Safe database file (`/etc/privdb/mls.db`). If the following lines are not already included in `/etc/privdb/mls.db` (and if your site has the `/usr/src/cmd/sp/privdb/mls.db` file), add them, as follows:

    ```
    file = { name = /etc/privdb/gen.db; MAC = [syslow,none]; };
    #if exists(/usr/spool/logs/airlog)
    file = { name = /usr/spool/logs/airlog; MAC = [syshigh,none]; };
    #endif
    ```

20. Run the `privcmd` command to label system files with the file privilege states, permission modes, owner, and group, and to assign privilege assignment lists (PALs) and security labels, as shown in the following example:

    ```
    /etc/privcmd
    ```

    Note: Any time you change (or add to) your system configuration, you must execute the `privcmd` command to label these files. This step is required for rebuilds also.

    If your current UNICOS system was not configured with the `syshigh` and `syslow` labels (`SECURE_MAC`), you will see error messages, such as `WARNING: syshigh/syslow MAC labels are not set.` Ignore these messages at this time.

21. Protect the NFS ID maps and the scripts that produce the maps (see page 1291, for more information).

    Every script or program used in ID map generation must have a `syslow` label. These scripts and programs must be executed at the `syslow` label, and the resulting map files must have a `syslow` label. The NFS commands in the `/etc/uidmaps` directory and the directory itself are automatically installed with the `syslow` label. Add the `syslow` label to locally-written scripts. When `cron`(8) is used to execute the scripts, ensure that the `cron` job is set to run with the `syslow` label. Existing ID map files are overwritten by the `nfsmerge`(8) command without changing their labels; you should remove all existing ID map files at the start of the ID map generation process.

22. Shut down the currently running UNICOS `PRIV_SU` system

23. Boot the Cray ML-Safe system configuration, and stay in single-user mode.

24. Disable auditing by executing the `/etc/spaudit -d state` command.

25. Execute the following sequence of commands:

    ```
    /etc/mount /usr
    /etc/mount /usr/src
    /etc/mount /usr/spool  # if the site has one
    /etc/mount /usr/adm    # if the site has one
    /etc/mount /usr/adm/sl # if the site has one
    /etc/privcmd
    ```

This sequence of commands must be executed because the various objects labeled by `privcmd` do not have the correct security label unless the `SECURE_MAC` configuration option is enabled.

26. Reenable auditing by executing the `/etc/spaudit -e state` command.

27. Unmount all file systems, execute the `sync` command, and reboot the Cray ML-Safe configuration, as follows:

    ```
    cd /
    /etc/umountem
    sync
    sync
    sync
    /etc/ldsync
    <reboot>
    ```

28. Go to multiuser mode by executing the `init 2` command.

8.7.9.2 Single level UNICOS system to a multilevel UNICOS system

Use the following procedure to convert your single-level UNICOS system to a multilevel UNICOS system:

1. Configure and build a UNICOS kernel. This can be done during normal production hours. Enter the menu system as follows:

   ```
   cd /etc/install
   ./install
   ```

2. To administer your UNICOS MLS configuration options using the UNICOS Installation and Configuration Menu System, enable it by setting the `Configure system->Configurator automation options->Security configuration option` to `YES`.

3. Import the default security level names by selecting `Configure system->Multilevel Security (MLS) configuration->Import the security configuration option` in the UNICOS Installation and Configuration Menu System.

4. Set up all new site-defined security level and compartment names at this time so you do not have to rebuild the kernel later. To define new security level names, use the `Configure system->Multilevel security (MLS) configuration->Site labels configuration->Levels` selection in the UNICOS Installation and Configuration Menu System. To define new security compartment names, use `Configure`

`system->Multilevel security (MLS) configuration->Site labels configuration->Compartments` selection in the UNICOS Installation and Configuration Menu System.

5. Set all the necessary configuration parameters found in the `Configure system->Multilevel security (MLS) configuration` menu in the UNICOS Installation and Configuration Menu System.

   One of the most important selections is the system management mechanism. Cray Research recommends enabling the `PRIV_SU` mechanism, which means the super-user policy is enforced. Enable this option by using the `Configure system->Multilevel security (MLS) configuration->System options->Super-user privilege policy?` selection in the UNICOS Installation and Configuration Menu System.

6. Configure the security levels and compartments on the networking interfaces found by using the `Configure system->Network configuration->General network configuration->Network interface` selections in the UNICOS Installation and Configuration Menu System.

   If these interfaces and routes are not labeled, the authorized security levels and compartments of the users connecting to the UNICOS system are set to 0 and none, respectively.

7. When all the configuration options have been set, build the kernel by activating the security and system configuration. Activating the configuration, which updates the appropriate source files, must be done before building the UNICOS kernel. To activate the configuration, use the `Configure system->Activate the configuration` selection in the UNICOS Installation and Configuration Menu System.

   Each time a configuration is activated, the Installation and Configuration Menu System determines which files are affected by the changes. The following message is then displayed:

   ```
   Do you want to proceed with the configuration update (y/n)?
   ```

   Respond by typing `y`.

8. Build the kernel. This involves three selections in the `Build/install system` selection of the UNICOS Installation and Configuration Menu System. First, use the `Build/install system->Components to build` selection and set to selected major components. Then use the `Build/install system->Components to build` selection to set all

selections to `NO`, except for the UNICOS kernel selection, which must be set to `YES`. Last, use the `Build/install system->Do the build` selection to rebuild the kernel selection.

9. Transfer the newly built UNICOS kernel to the boot medium.

10. Label all file systems that are to be mounted under the UNICOS kernel. At a minimum, these file systems must be labeled by using the `labelit`(8) command, as shown in the following example:

    ```
    /etc/labelit -s -u 0 -l 0 -c 0 /dev/dsk/user1
    ```

    Note: You must use the `-u`, `-l`, and `-c` options when using the `-s` option of the `labelit` command.

    If your site uses nonzero levels and compartments, then use the following guidelines for labeling file systems:

    Note: If `SECURE_MAC` is enabled, then the range of labels should be `syslow` to `syshigh`.

    - The `root` file system must be labeled with all the security levels and compartments that are assigned to users on the UNICOS system. This is necessary because pipes are usually allocated through the `root` device. Also, tty devices change labels to match that of the user through the `root` device.
    - The file system that contains the NQS, mail files, and line printer queues must be labeled with all the security levels and compartments assigned to users on the UNICOS system.
    - The `/usr/tmp` and `/tmp` files must also be labeled with all the security levels and compartments that are assigned to users on the UNICOS system.
    - If the system is configured with `SECURE_MAC` is enabled and `syslow` through `syshigh` security labels, you must ensure the core file system (where dumps are to be written) is authorized for `syshigh`.
    - If the system is configured with `SECURE_MAC` enabled and `syslow` through `syshigh` security labels, you must ensure the job temporary file system (if it differs from `/tmp`) is authorized for `syslow`.

11. Your site may need to specify a maximum security level and authorized compartments values that are nonzero. For example, if your site plans on

using security levels 0 through 16 and compartment 04700, then your `root` file system must be labeled as follows:

```
/etc/labelit -s -l 0 -u 16 -c 04700 /dev/dsk/root
```

Another example is if the system is configured with `SECURE_MAC` enabled, and `syslow` through `syshigh` security labels, then the `root` file system must be labeled as follows:

```
/etc/labelit -s -l syslow -u syshigh -c 07777 /dev/dsk/root
```

You cannot label the `root` file system while mounted, which means you must boot an alternate `root` file device. Label the original `root` file system as previously described in this procedure and boot this labeled `root` file system.

12. All user file systems must be correctly labeled, as shown in the following example:

    ```
    /etc/labelit -s -l 0 -u 3 -c 01000 /dev/dsk/user1
    ```

    Boot the new UNICOS kernel into single-user mode. For more information about booting into single-user mode, see the installation guide for your CRAY system.

13. Disable auditing in single-user mode as shown in the following example:

    ```
    /etc/spaudit -d state
    ```

    If you run a `PRIV_SU` and PALs system, then execute the `privcmd` command, as shown in the following example:

    ```
    /etc/privcmd
    ```

14. Unmount all file systems mounted in the previous step.

15. Create or update a UDB login for the security administrator. See Section 8.7.6, page 239, for more information on setting up this entry. Depending on the privilege mechanism(s) used, the security administrator login should be given the additional UDB fields with the following values:

    - For `PRIV_SU` systems: `permits:suidgid:;`
    - For systems with PALs: `valcat:secadm:`

Ensure that the `chown` permbit is assigned to the security administrator account, as shown in the following example:

```
/etc/udbgen
udbgen: 1>update:user:
udbgen: 2>valcat:secadm:maxcls:0:
udbgen: 3>permits:suidgid:
udbgen: 4>permbits+:chown:
udbgen: 5>quit
Updated 1 record
```

Additionally, if the security administrator is to reclassify user files, assign all the security levels and compartments for the system to the security administrator account.

16. For UNICOS systems, the system console device entries under the `/dev` directory must be labeled. The `getty -L` and `getty -C` commands specify the security level and compartment ranges, respectively, for the system console devices, `/dev/console`, `/dev/tty00`, `/dev/tty01`, and so on. Additionally, you can use the `getty -m` command to indicate that the specified console device is a multilevel device.

    If `SECURE_MAC` is enabled, label the console to `syslow`.

    The `getty` command for these devices are specified in the `/etc/inittab` file, as shown in the following example:

```
co::respawn:/etc/getty -L 0-16 -C 0-0777777777777777777777 console console
01:23:respawn:/etc/getty -L 0-16 tty01 9600
02:23:respawn:/etc/getty -L 0-5 tty02 9600
```

17. If TCP/IP network access is to be permitted to the UNICOS system (for example, `telnet` or `ftp`), entries for the following Internet address(es) must appear in the network access list (NAL):

    - Localhost (`127.0.0.1`)
    - Local network interfaces for all directly-connected networks
    - Network interfaces on the OWS (if network access is permitted to the UNICOS system from the OWS-E).

18. On a UNICOS system, a default NAL entry controls the label given to a host not specified in the kernel NAL table. If a default entry is not in the NAL, then only specified hosts are allowed to connect to the UNICOS system. Add the default entry to the NAL if you want hosts not specified

in the kernel NAL table to connect to the UNICOS system. The default entry can be added by using the `Configure System->Multilevel Security (MLS) Configuration->Network Security Options->Network Protocol Security Configuration->MLS Network Access List (NAL) Sets and MLS Network Security Definitions` selection in the UNICOS Installation and Configuration Menu System. The default entry host name to use is `default`.

The following is the NAL definition from the file `/etc/config/spnet.conf`, which is generated by the Installation and Configuration Menu System. In this example, the default entry is allowed only a nonzero label connection.

```
ip net "default" {
    name = "%NBY";
    class = B1;
    max label = level0, 0;
}
```

See `spnet`(8) for more information on setting up the NAL.

19. If used, the workstation access list (WAL) entries for the UNICOS system should also be generated. The WAL is not required and may be configured at a later time. To configure the WAL entries at this time, use the `Workstation Access List (WAL) Sets` selections in the UNICOS Installation and Configuration Menu System. See `spnet`(8) for more information on setting up the WAL.

20. If the local networks on the site support the Commercial IP Security Option (CIPSO), you must create the security label translation tables. The `CIPSO Map Domain Sets` selection of the UNICOS Installation and Configuration Menu System defines the maps, and the `Network Security Definitions` selections associate CIPSO hosts with their CIPSO maps. See `spnet`(8) for more information on setting up the CIPSO maps.

21. The security administrator must also define each user in the UDB. Each UDB entry defines the active and authorized security levels, active category, authorized categories, active and authorized compartments, and permissions. Proper definition of a user in the UDB is critical to maintaining a Cray ML-Safe configuration of the UNICOS system. Ensure that each user is given only those levels, categories, compartments, and permissions that are absolutely necessary.

    The security administrator can use the `nu`(8) or `udbgen`(8) commands to create or modify UDB entries. (The `udbgen` command does not perform all the initialization tasks that the `nu` command performs.) This can be done in single-user mode, but to do so successfully, mount all file systems that the `nu` command needs to create the user login directories. If mounted, these file systems should be unmounted before going to the next step.

It is not recommended that the system be put in multiuser mode at this point, but bringing up multiuser mode without the network is an alternative to running the `nu` command in single-user mode.

Set proper security labels on the home directories of users with a nonzero default level and/or a nonnull default compartment set. This can be done by using the `spset -l` and `spset -c` command.

The `nu` command creates and initializes a home directory for each new user, but it cannot set it to a nonzero label. If users try to log in they will not be able to access their home directory unless it has been changed from its nonzero status. Any other files associated with the user's home directory may also require new label settings if the user is allowed to access them (for example, `.cshrc` or `.profile`). Files created after a user has logged in are automatically set with the correct security label.

22. Determine if MLDs are to be used and which directories must be converted to MLDs. Sites can use all wildcards, all MLDs or a mix of both. If `mail` is to be used in a nonzero label environment, convert the `/usr/mail` and `/usr/spool/mqueue` directories to MLDs.

    If `cron`(8), `at`(1), or `lpr`(1) are to be used in a nonzero label environment, then their corresponding spool directories must be set up as MLDs. See Section 8.4.1.2.2, page 177, for more information. For information on using MLDs for NQS, see *NQE Administration*, Cray Research publication SG–2150.

23. Prior to booting to multiuser mode, reenable auditing, as shown in the following example:

    ```
    /etc/spaudit -e state
    ```

    Go to multiuser mode by doing the following:

    ```
    /etc/init 2
    ```

    After entering multiuser mode, you can change the UNICOS MLS configuration parameters, but the kernel must be rebuilt and rebooted after the changes are made.

8.8 MLS auditing on a UNICOS system

MLS auditing on a UNICOS system consists of collecting data on security-relevant events. As shown in Figure 10, the kernel determines whether an auditable event should be audited based on the audit selection criteria. These criteria are saved in the low memory table in the kernel. The initial settings of the criteria are defined by the configuration parameters, which are set by the UNICOS Installation and Configuration Menu System; these settings can be changed by using either the UNICOS Installation and Configuration Menu System or `spaudit`(8). If an event is being audited, when the event occurs, a record is written to the security log pseudo device, `/dev/slog` (see `slog`(4)).

The security logging daemon (`slogdemon`(8)) reads `/dev/slog` and writes the records to the disk-resident security log (`/usr/adm/sl/slogfile`). The security administrator can generate (by using the `reduce`(8) command) the audit records from the security log to produce information on how the Cray ML-Safe configuration of the UNICOS system is being used.

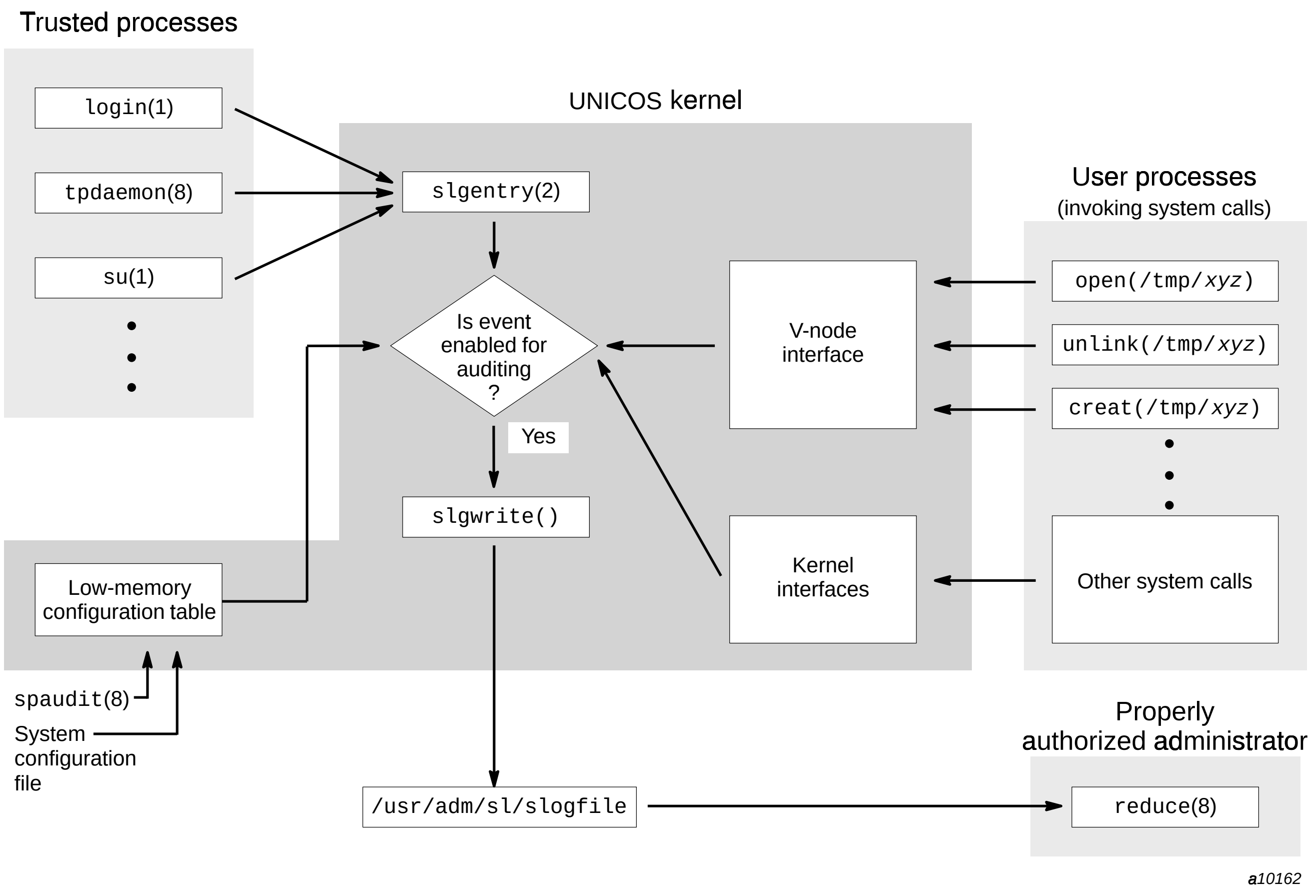

Figure 10. Overview of security auditing

For UNICOS systems using the PAL-based privilege mechanism, the security log is protected with a `syshigh` label and you must be an authorized user with the `exec` privilege text to use the `reduce` command to access the log.

The UNICOS system maintains a lockout parameter that is common to the security log device driver to ensure activation of only a single security log file and daemon.

The following sections explain auditing by providing the following information:

- Description of the security log and how to enable and configure it

- Description of the security log daemon
- How to enable the type of events to be recorded in the log
- Description of the security log record types
- Examples of how to use the `reduce` command

8.8.1 Security log overview

The `/etc/slogdemon` (see `slogdemon`(8)) program is the system security logging daemon. The daemon collects security log records from the operating system by reading the security log pseudo device, `/dev/slog` (see `slog`(4)), and writing the records to a disk-resident security log (`/usr/adm/sl/slogfile`).

`/dev/slog` is a read-only pseudo device that buffers security log records. By default, it holds approximately 1000 security log records (see the description of `SLG_BUFSIZE` later in this section for more information on defining the size of `/dev/slog`).

If `/dev/slog` becomes more than 50% full, each process that tries to generate audit records is tested to see if it matches one of the following conditions:

- If the buffer is more than 50% full and `slogdemon` is not running, all nonadministrative processes are put to sleep.
- If the buffer is more than 87% full, all nonadministrative processes are put to sleep.
- If the buffer is 93% full, all processes, except for the security log daemon and the idle process are put to sleep.

Once the buffer is emptied, a wakeup is sent to the sleeping processes.

If standard system buffering is done through `ldcache`, and the system panics, the flush-on-panic routine, if it is enabled, attempts to flush the `ldcache` buffers and system buffers to disk. Repeated attempts are made to flush buffers to disk. Because of this flushing mechanism, the potential for losing records is decreased, but records can still be lost under the following set of circumstances:

- When records are being transferred into a system buffer by a `write`(2) system call.
- When records are being transferred out of a system buffer.

The situation that provides the greatest chance for losing records is when the records reside in `/dev/slog` and the flush-on-panic routine is not enabled. The number of lost records depends on the size of `/dev/slog`. If the default setting is used, then approximately 1000 records would be lost. `/dev/slog` can be read from a system dump by using the `slog` and `rslog` commands of `crash`(8).

If the security logging daemon is not running, `/dev/slog` eventually fills up, and all processes that require security log entries sleep while waiting for the device to be emptied by the daemon. The amount of processing that can be accomplished when the daemon is not active depends upon the size of `/dev/slog` and the volume of security log data being sent to it. Even if the security log daemon is running, the size of `/dev/slog` can be configured too small and a system panic or hang can result.

The parameters that define the state, location, and size of the security log are found in `uts/cf.SN/config.h`, and are as follows:

Parameter	Description
`SLG_STATE`	Defines if the security log is enabled. By default, the security log is `ON`.
`SLG_DIR`	Defines the disk-resident security log directory for both active and retired security logs. The default is `/usr/adm/sl`.
`SLG_FILE`	Defines the file name, relative to `SLG_DIR`, of the active disk-resident security log file (that is, `SLG_DIR/SLG_FILE` defines the file name for the active security log file). The default is `slogfile`.
`SLG_FPREFIX`	Defines the file name prefix, relative to `SLG_DIR`, of the retired disk-resident security log file (that is, `SLG_DIR/SLG_FPREFIX` defines the file name prefix for the retired security log files). The default is `s`.
`SLG_MAXSIZE`	Defines the threshold value, which once reached, causes the disk-resident security log to be retired. The default is 8,192,000 bytes.
`SLG_BUFSIZE`	Defines the size of `/dev/slog`; the default is 163,840 bytes.

The `SLG_STATE` parameter allows you to enable or disable the security log. If disabled, all other security logging options are ignored. To set this parameter, use the `Configure system->Multilevel security (MLS)` `configuration->Security log file configuration->Enable security logging?` selection in the UNICOS Installation and Configuration Menu System.

The `SLG_DIR` parameter allows you to specify the directory in which both active and retired disk-resident security logs are kept. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Directory for security logs` selection in the UNICOS Installation and Configuration Menu System.

The `SLG_FILE` parameter allows you to specify the actual name (within the directory specified by `SLG_DIR`) of the active disk-resident security log. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Filename of active security log` selection in the UNICOS Installation and Configuration Menu System.

The `SLG_FPREFIX` parameter allows you to define the file name prefix (within the directory specified by `SLG_DIR`) of a disk-resident security log file that is archived. The path to the archived security log file must reside on the same file system as `SLG_FILE`. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Prefix for retired security logs` selection in the UNICOS Installation and Configuration Menu System.

The `SLG_MAXSIZE` parameter defines the threshold value (in bytes), that once reached, causes the disk-resident security log to be retired. The actual maximum size of the security log may be larger than `SLG_MAXSIZE` by a small amount (less than 1024 bytes). It is recommended to make `SLG_MAXSIZE` no smaller than the default; increments should be made in 4096-word blocks.

When the log reaches `SLG_MAXSIZE`, it is renamed with a `SLG_DIR/SLG_FPREFIX`*yymmddhhmmss* name, where *yymmddhhmmss* is the time of the renaming, and a new log file is created. The security/system administrator must archive the renamed logs promptly to prevent the file system containing the security log from becoming full. If the system runs out of disk space for the security log, `/dev/slog` eventually fills, causing the system to stop.

To set the `SLG_MAXSIZE` parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Maximum log file size` selection in the UNICOS Installation and Configuration Menu System menu.

The SLG_BUFSIZE parameter defines the size of /dev/slog. The default size is 163,840 bytes (approximately 1000 security log records). The default size should not be decreased unless your site has never enabled auditing. In this case, SLG_BUFSIZE can be set to 0. If the SLG_ALL_NAMI or SLG_ALL_VALID configuration parameters are enabled, the default should be increased by a factor of two or three. Any increase to the default should be made in 8-byte increments (that is, multiples of 8).

To set SLG_BUFSIZE, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log buffer size selection in the UNICOS Installation and Configuration Menu System.

8.8.2 Security logging daemon

The security logging daemon is capable of running in all states (that is, single-user mode, multiuser mode, and so on). The default is for the daemon to run only in multiuser mode.

The security logging daemon should be initiated through inittab(5) or the /etc/rc script. At installation time, the install scripts create the pseudo device for /dev/slog. If it is necessary to recreate it, refer to the mknod(8) man page. The major device number is 20 and the minor device number is 0 for /dev/slog.

To restart slogdemon in multiuser mode, locate the process ID of the slogdemon process. Then, send a termination signal (15) to the process and restart the daemon, as shown in the following example:

```
kill -15 ‘cat /etc/slogd.pid‘ && sleep 1 && /etc/slogdemon
```

If the daemon is started manually, then it must be done by root (UID = 0). The security administrator and appropriate operators should be the only users who know the password for this login ID.

8.8.3 Security logging daemon in single-user mode

There are two methods to initiate the security logging daemon in single-user mode. The first method is to add a command to inittab to mount all file systems in the path to the security log file and then start the daemon. This is shown in the following example. In this example, a separate sl file system contains the security log; where the files are actually placed is site-dependent:

```
slg::sysinit:/etc/mount /dev/dsk/sl /etc/sl&&/etc/slogdemon #start slog
```

Ensure that the path to the security log in `inittab` is the same as specified by the security log configuration parameter in `uts/cf.SN/config.h`. To do this, set the `SLG_DIR` parameter by using the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Directory for security logs` selection in the UNICOS Installation and Configuration Menu System.

The second method is to start the daemon manually. To do this, you must first mount the file system and then start `slogdemon`. Before going to multiuser mode, you must kill `slogdemon` and unmount the file system. If you restart the kernel in single-user mode, you must first kill `slogdemon` and unmount the file system.

8.8.4 The `spaudit(8)` command

The `spaudit`(8) command allows a properly authorized administrator to change the security auditing criteria while the UNICOS system is running. This command can also be used to change the security log edition.

The `spaudit -e` *enableopts* command enables the specified security logging configuration option(s). The `spaudit -d` *disableopts* command disables the specified security logging configuration option(s). The valid options are as follows:

Option	Description
`all_nami`	Logs all `mkdir`, `rmdir`, `link`, and `rm` calls
`all_rm`	Logs all remove requests
`all_valid`	Logs all access requests
`audit`	Logs all security auditing criteria changes
`chdir`	Logs all change directory requests
`config`	Logs all UNICOS configuration changes
`crl`	Logs Cray/REELlibrarian activity
`dac`	Logs discretionary access changes
`discv`	Logs discretionary access violations
`filexfr`	Logs all file transfer requests
`ipnet`	Logs all network violations
`jend`	Logs end of job
`jstart`	Logs start of job

`linkv`	Logs all link (`ln`) violations
`mandv`	Logs mandatory access violations
`mkdirv`	Logs all make directory (`mkdir`) violations
`netcf`	Logs network configuration changes
`netwv`	Logs network violations (not currently used)
`nfs`	Logs all NFS activity
`nqs`	Logs NQS activity
`nqscf`	Logs NQS configuration changes
`object_path`	Logs object's full path name on accesses
`operator`	Logs operator actions
`priv`	Logs use of privilege
`removev`	Logs all remove violations
`rmdirv`	Logs all remove directory (`rmdir`) violations
`secsys`	Logs all security system call requests
`setuid`	Logs all setuid requests
`shutdown`	Logs system shutdown requests
`startup`	Logs system startup requests
`state`	Defines if security log is on (`1`) or off (`0`)
`sulog`	Logs all `su` attempts
`tapes`	Logs tape activity
`time_change`	Logs system time change
`trust`	Logs Cray ML-Safe process activity
`user`	Logs user password for failed login attempts

The command can be used in either single-user or multiuser mode to enable/disable security logging options. See the `spaudit`(8) man page for more information on using this command.

8.8.5 Security logging configuration parameters

The auditing of all security-relevant events can be configured on a UNICOS system. The following list summarizes the configuration parameters that enable/disable the logging of security events:

Parameter	Description (default setting)
`SLG_DISCV`	Logs discretionary access violations (on)
`SLG_MANDV`	Logs mandatory access violations (on)
`SLG_NETWV`	Currently not used
`SLG_MKDIRV`	Logs all make directory (`mkdir`) violations (on)
`SLG_RMDIRV`	Logs all remove directory (`rmdir`) violations (on)
`SLG_LINKV`	Logs all link (`ln`) violations (on)
`SLG_ALL_RM`	Logs all remove (`rm`) requests (off)
`SLG_REMOVEV`	Logs all remove violations (on)
`SLG_ALL_NAMI`	Logs all `mkdir`, `rmdir`, `link`, and `rm` calls (off)
`SLG_ALL_VALID`	Logs all access requests (off)
`SLG_ALL_NETW`	Currently not used
`SLG_PHYSIO_ERRORS`	Currently not used
`SLG_CF_NET`	Logs all network configuration changes (off)
`SLG_FILEXFR`	Logs all file transfers (off)
`SLG_PATH_TRACK`	Tracks all path names on accesses (on)
`SLG_SULOG`	Logs all `su` attempts (on)
`SLG_NFS`	Logs all Cray-to-Cray NFS requests (off)
`SLG_CF_UNICOS`	Logs UNICOS configuration changes (off)
`SLG_CF_NQS`	Logs NQS configuration changes (off)
`SLG_JSTART`	Logs start of job (on)
`SLG_JEND`	Logs end of job (on)
`SLG_SUID_RQ`	Logs all setuid requests (on)
`SLG_USER`	Logs name and password on login password failure (off)
`SLG_ACT_NQS`	Logs NQS activity (off)
`SLG_T_PROC`	Logs Cray ML-Safe process activity (off)
`SLG_LOG_PRIV`	Logs use of privilege in system calls (off)
`SLG_LOG_AUDIT`	Logs all changes of audit criteria (off)
`SLG_LOG_CHDIR`	Logs all change directory requests (off)

`SLG_LOG_CRL`	Logs Cray/REELlibrarian activity (off)
`SLG_LOG_DAC`	Logs discretionary access changes (on)
`SLG_LOG_IPNET`	Logs all network violations
`SLG_LOG_OPER`	Logs operator actions (off)
`SLG_LOG_SECSYS`	Logs security system calls (off)
`SLG_LOG_STARTUP`	Logs system startup (off)
`SLG_LOG_SHUTDWN`	Logs system shutdown (off)
`SLG_LOG_TAPE`	Logs tape activity (off)
`SLG_LOG_TCHG`	Logs system time change (off)

8.8.6 Security log record types

There are different records that can be logged when the security log is enabled (`SLG_STATE = ON`). The types of records are summarized in the following list:

- System start record (`SLG_GO`)
- System logging stop record (`SLG_STOP`)
- System configuration change record (`SLG_CCHG`)
- System time change record (`SLG_TCHG`)
- Discretionary access violation record (`SLG_DISC_7`)
- Mandatory access record (`SLG_MAND_7`)
- Operational access record (`SLG_OPER`)
- Login validation record (`SLG_LOGN`)
- Tape activity record (`SLG_TAPE`)
- End-of-job record (`SLG_EOJ`)
- Change directory record (`SLG_CHDIR`)
- Security system call record (`SLG_SECSYS`)
- Discretionary access change record (`SLG_DAC_CHNG`)
- `setuid` system call record (`SLG_SETUID`)
- `su` attempt record (`SLG_SU`)

- File transfer logging record (`SLG_FXFR`)
- Network security violations record (`SLG_IPNET`)
- Cray NFS request record (`SLG_NFS`)
- Network configuration change record (`SLG_NETCF`)
- Audit criteria selection change record (`SLG_AUDIT`)
- NQS configuration change record (`SLG_NQSCF`)
- NQS activity record (`SLG_NQS`)
- Cray ML-Safe process activity record (`SLG_TRUST`)
- Use of privilege record (`SLG_PRIV`)
- Cray/REELlibrarian activity record (`SLG_CRL`)

Table 10 summarizes the record types, the configuration parameter that enables the generation of the record, the `spaudit` argument that enables generation of the record, and the `reduce` option used to view the record.

Table 10. Security log records

Record type	Configuration parameter	spaudit -e	reduce -t
SLG_GO	SLG_LOG_STARTUP	startup	go
SLG_STOP	SLG_LOG_SHUTDWN	shutdown	stop
SLG_CCHG	SLG_CF_UNICOS	config	cchg
SLG_TCHG	SLG_LOG_TCHG	time_change	tchg
SLG_DISC_7	SLG_DISCV	discv	disc
SLG_MAND_7	SLG_MANDV and/or SLG_ALL_VALID	mandv, all_valid	
SLG_OPER	SLG_LOG_OPER	operator	oper
SLG_LOGN	SLG_JSTART	jstart	logn
SLG_TAPE	SLG_LOG_TAPE	tapes	tape
SLG_EOJ	SLG_EOJ	jend	eoj
SLG_CHDIR	SLG_PATH_TRACK and SLG_LOG_CHDIR	object_path, chdir	chdir
SLG_SECSYS	SLG_LOG_SECSYS	secsys	secsys
SLG_NAMI	SLG_MKDIR, SLG_RMDIRV, SLG_LINKV, SLG_REMOVEV, SLG_ALL_RM, and/or SLG_ALL_NAMI	mkdirv, rmdirv, linkv, removev, all_rm, all_nami	nami
SLG_DAC_CHNG	SLG_LOG_DAC	dac	dac
SLG_SETUID	SLG_SUID_RQ	setuid	setuid
SLG_SU	SLG_SULOG	sulog	sulog
SLG_FXFR	SLG_FILEXFR	filexfr	xfer
SLG_IPNET	SLG_LOG_IPNET	ipnet	netip
SLG_NFS	SLG_NFS	nfs	nfs
SLG_NETCF	SLG_CF_NET	netcf	netcf
SLG_AUDIT	SLG_LOG_AUDIT	audit	audit
SLG_NQSCF	SLG_CF_NQS	nqscf	nqscf
SLG_NQS	SLG_ACT_NQS	nqs	nqs

Record type	Configuration parameter	`spaudit -e`	`reduce -t`
`SLG_TRUST`	`SLG_T_PROC`	`trust`	`trust`
`SLG_PRIV`	`SLG_LOG_PRIV`	`priv`	`priv`
`SLG_CRL`	`SLG_LOG_CRL`	`crl`	`crl`

Assigning the `usrtrap` permission to a user or the `trapr` or `trapw` flags to an object overrides any security logging configuration parameter setting except for `SLG_STATE`.

The `SLG_PATH_TRACK` parameter allows the logging of all path names on access requests. No specific record is used when this parameter is enabled, but instead it forces more data to be stored with each access type record generated by the other parameters. See Section 8.8.7.6, page 280, for an example of the information produced (see the example in which the `-p` option is used). This parameter must be enabled in order for the `reduce -p` command to work correctly.

To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Track all pathnames on accesses?` selection in the UNICOS Installation and Configuration Menu System.

The use of this parameter means that all `chdir`(2) requests must also be logged. See Section 8.8.7.12, page 303, for more information on enabling this record type.

If object path tracking is required for auditing of file manipulations that are being done by file descriptors, the `SLG_ALL_VALID` parameter must be enabled. If this parameter is enabled, then the initial file access that is made through the `open`(2) system call is logged and the object path name is included in this record. This allows the `reduce` utility to determine the full path name for any succeeding file descriptor operations that occur on this file.

For more information on enabling the `SLG_ALL_VALID` parameter, see Section 8.8.7.8, page 289.

8.8.7 Auditing on a Cray ML-Safe system configuration

The security auditing policy for a site running a Cray ML-Safe configuration can be defined by the site. That is, you can run a Cray ML-Safe configuration with or without security auditing enabled. It is recommended that the site policy be determined before the system is up and running and that it be applied consistently at all times. Consistent and proper use of the MLS auditing features helps ensure site security.

Also, there are no settings specified for the the security logging options on a Cray ML-Safe configuration, with the one exception noted in the next paragraph. Cray Research ships the UNICOS release with default settings for the security logging options. See Section 8.8.5, page 268, for information on the default settings.

To fulfill TCSEC auditing requirements, you must be able to identify an object by its full path name. A Cray ML-Safe configuration can meet this requirement only if the path-tracking configuration option (`SLG_PATH_TRACK`) is enabled. This option is on by default.

8.8.7.1 Security log record header definition

Each security log record has a header (`slg_hdr`), followed by record-specific information. The kernel provides the header information for all kernel-generated security log records. A non-kernel Cray ML-Safe process that is performing an activity on behalf of a user formats its own security log record header. However, the kernel also places information in the header fields for the record.

To meet TCSEC requirements regarding the ability to select records by the object label, it was necessary to expand the number of fields in the security log record header.

To maintain compatibility with earlier releases of UNICOS with the MLS feature, the older version of the security log header (`slghdr0`) is retained and a new header structure (`slghdr`) that contains the same information as `slghdr0` is provided, plus the fields necessary to record the security label information (that is, both the security level and compartments) of the subject and the object. The older version of the header contains only the security level of the subject and object.

The `reduce` command uses the `-S` and `-L` options, which allow you to select either the new version (`-S`) or older version (`-L`) of header information in the display produced by the `reduce` command.

The use of the `-S` option is shown in some of the examples in the following sections. If the `-S` or `-L` options are not specified, the older version of the security log header is displayed by `reduce` (that is, the `-L` display is the default for the UNICOS system).

See Section 8.8.8.2, page 345, for more information on these options.

The security log record header using the older version is as follows:

```
Date_time            Type       o_lvl:n   s_lvl:n   jid:n  pid:n
  r_ids:[ssss(n),ssss(n)]      e_ids:[ssss(n),ssss(n)]       **********
Login uid: ssss(n)
```

The new version of the security log record header is as follows:

```
Date_time   Type   jid:n   pid:n   r_ids:[ssss(n),ssss(n)]
e_ids:[ssss(n),ssss(n)]
    S_Label: level,cmpts  O_Label: level,cmpts
    Login uid: ssss(n)
```

The fields are defined as follows:

Field	Description
`Date_time`	The date and time at which the kernel entered the record in `/dev/slog`.
`Type`	The type-of-record.
`o_lvl:n`	The security level of the object; this is not meaningful if no object is involved. This field is used only on the older version of the record header.
`s_lvl:n`	The security level of the subject. This field is used only on the older version of the record header.
`jid:n`	The job ID.
`pid:n`	The process ID.
`r_ids:[ssss(n),ssss(n)]`	The real user and group names and IDs in the form of [*username(user ID),groupname(group ID)*].

`e_ids:[ssss(n),ssss(n)]`

The effective user and group names and IDs in the form of [*username(user ID),groupname(group ID)*].

`S_Label:  level,cmpts`

The security label of the subject. This field is used only on the new version of the record header.

`O_Label:  level,cmpts`

The security label of the object. This field is used only on the new version of the record header.

`Login uid:  ssss(n)`

The user name and ID at the time of job initiation in the form of *username(user ID*). This is the most reliable identifier in the log entry because it cannot be changed after logging in. This ID is the one for which the password was checked. The login user ID is in the initial `SLG_LOGN` record for any job, regardless of origin. The only exception to this occurs when a user executes the `setsid`(2) or `setjob`(2) system call to create a new session. In this case, the login user ID of the new session is shown in the `SLG_SECSYS` record, that is generated by the `setsid` or `setjob` system call.

The format of the record-specific information following the header can differ, although some records have the same output format, as described in the following sections.

Note: Some of the security audit records described in the following sections use a `Class` field. This field is no longer used and its contents are not meaningful. The class value should be set to 0 by a properly authorized user, as nonzero values are no longer used on UNICOS systems.

8.8.7.2 System start record (`SLG_GO`)

The `SLG_GO` record is written as the system is booted into single-user mode. The `SLG_LOG_STARTUP` parameter must be on for this record to be generated. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log system startup?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e startup` command to enable the generation of this record or the `spaudit -d startup` command to disable the generation of this record.

In addition to the header information, the `SLG_GO` record displays the following information:

```
System Node  Release Version  Machine Mem: Avail_user_mem/Max_mem
```

The fields are defined as follows:

Field	Description
`System`	Value of the `SYS` parameter in `utsname`
`Node`	Value of `NODE` parameter in `utsname`
`Release`	Release level of operating system
`Version`	Version level of operating system
`Machine`	Type of machine
`Mem:  Avail_user_mem/Max_mem`	The amount of available user memory and configured memory, respectively

To display a `SLG_GO` record, use the `reduce` command, as shown in the following example.

```
$ /etc/reduce -t go
Feb  5 17:00:00 1992  Startup     o_lvl: 0  s_lvl: 0  jid:0  pid:0
  r_ids:[root(0),root(0)]      e_ids:[root(0),root(0)]       **********
Login uid: root(0)

  sn1405 sn1405 8.0.0ah d80.55 CRAY Y-MP Mem: 32595968/33554176
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.3 System shutdown record (`SLG_STOP`)

The `SLG_STOP` record is written when the security log daemon receives a `SIG_TERM` signal, causing the daemon to exit. This occurs only when an operator issues the `shutdown`(8) command. The only information recorded in this record is the header information described in Section 8.8.7.1, page 274, for more information.

The `SLG_LOG_SHUTDWN` parameter must be on for this record to be generated. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log system shutdown?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e shutdown` command to enable the generation of this record or the `spaudit -d shutdown` command to disable the generation of this record.

8.8.7.4 System configuration change record (`SLG_CCHG`)

The `SLG_CCHG` record is written if the system's configuration is changed by using the `udbgen`(8), `nu`(1), or `xadmin`(8) commands.

The `SLG_CF_UNICOS` parameter must be on for this record to be generated. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log UNICOS configuration changes?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e config` command to enable the generation of this record or the `spaudit -d config` command to disable the generation of this record.

To display a `SLG_CCHG` record, use the `reduce` command, as shown in the following example. The fields in this record type are self-explanatory:

```
$ /etc/reduce -t cchg
Jul 15 10:34:19 1993 Configuration o_lvl: 0 s_lvl: 0 jid:1231 pid:33951
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
    Login uid: operator(9)

                   Subtype: UDB change
           login directory: /
                login root: /
               login shell: /bin/sh
                login name: operator
                    action: changed
                       uid: 9
                valid gids: 9
                  permbits:
            login failures: 0
             default level: level0
                 max level: level0
                 min level: level0
          default comparts: none
            valid comparts: none
            authorizations: none
                     flags: none
        default categories: none
          valid categories: sysops
                  password: same
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.5 System time change record (SLG_TCHG)

The SLG_TCHG record is written when the system's time is set during the system startup sequence. This record is also generated when an administrator changes the system's time with the date(1) command.

The SLG_LOG_TCHG parameter must be on for this record to be generated. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log system time change? selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e time_change` command to enable the generation of this record or the `spaudit -d time_change` command to disable the generation of this record.

In addition to the header information, the `SLG_TCHG` record displays the following information:

```
Time changed to: time
```

This information displays the system's new time.

To display a `SLG_TCHG` record, use the `reduce` command, as shown in the following example:

```
$ /etc/reduce -t tchg
Apr  1 09:09:38 1991  Time        o_lvl: 0  s_lvl: 0  jid:2  pid:4
  r_ids:[root(0),root(0)]      e_ids:[root(0),root(0)]       **********
Login uid: root(0)

  Time changed to: Mon Apr  1 09:09:22 1991
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.6 Discretionary access violation record (`SLG_DISC_7`)

The discretionary access violation record is written for all discretionary access failures. The `SLG_DISC_7` record is used.

The `SLG_DISCV` parameter must be on for this record to be generated. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log discretionary access violations?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e discv` command to enable the generation of this record or the `spaudit -d discv` command to disable the generation of this record.

A discretionary access violation record is generated for any discretionary access violation that occurs when a process attempts any operation that involves evaluation of access to a file system object or its path.

Specifically, discretionary access violation records can be generated for the following system calls: `access`(2), `acct`(2), `chacid`(2), `chdir`(2), `chmod`(2), `chown`(2), `chroot`(2), `creat`(2), `dacct`(2), `exec`(2), `getacl` (2), `jacct`(2), `join`(2), `lchown`(2), `link`(2), `lsecstat`(2), `lstat`(2), `mkdir`(2), `mknod`(2), `mount`(2), `msgsys` (2), `open`(2), `pathconf`(2), `readlink`(2), `rename`(2), `rmdir`(2), `rmfacl`(2), `secstat`(2), `semsys` (2) `setacl` (2), `setfcmp`(2), `setflvl`(2), `setpal`(2), `shmsys` (2), `stat`(2), `statfs`(2), `symlink`(2), `unlink`(2), and `utime`(2).

In addition to the header information, the `SLG_DISC_7` record displays the following information:

```
Function: func (nn)   Violation: text (nnn)
           System call: syscall(number)
Subject: Compartments : sub_comps
          Permissions : sub_permits
                Class : sub_intcls
           Categories : sub_intcat
          Access Mode : access_mode
Object: Level:lvl  uid: user(ID)  gid: group(ID)  device: maj,min
inode: INUM
             Pathname : file
         Compartments : obj_comps
                Class : obj_intcls
           Categories : obj_intcat
                 Mode : obj_mode
```

The fields are defined as follows:

Field	Description
`Function: func (nn)`	The function name and its system call number.
`Violation: text (nnn)`	The error message associated with this violation followed by the error number. If no violation occurred, the value `NONE(0)` is shown. See Section 8.8.10, page 354, for more information.
`System call: syscall(number)`	The system call and its number. The system call is printed after the function. This field is included, as the function that causes the discretionary access violation can be different than the system call.

`Subject: sub_comps`

The subject's active security compartments.

`Permissions: sub_permits`

The subject's authorized permissions (as defined in the UDB).

`Class: sub_intcls`

The subject's active class. This value is no longer used.

`Categories: sub_intcat`

The subject's active category.

`Access Mode: access_mode`

The subject's requested access mode for the desired object. This field is available only on 7.0 and later UNICOS systems with the MLS feature.

`Object: Level: lvl`

The object's security level.

`uid: user(ID)`

Owner of the object and the owner's user ID number.

`gid: group(ID)`

Owning group of the object and the owning group's group ID number.

`device: maj,min`

Major and minor device numbers of the device the object resides on.

`inode: INUM`

The inode number of the object.

`Pathname: file`

The full pathname to the object (displayed if the `-p` option of the `reduce` command is used and the path tracking option is enabled).

`Compartments: obj_comps`

The object's security compartments.

`Class: obj_intcls`

The object's integrity class. This value is no longer used.

`Categories: obj_intcat`

The object's category.

`Mode: obj_mode`

The object's UNICOS mode permission bits.

To display a discretionary access violation record, use the `reduce` command, as shown in the following example.

```
$ /etc/reduce -t disc
Apr  1 09:50:00 1991  Discretionary o_lvl: 0 s_lvl: 0 jid:167  pid:7026
  r_ids:[operator(9),operator(9)] e_ids:[operator(9),operator(9)]
**********
Login uid: operator(9)

  Function: open (5)     Violation: Permission denied (13)
            System call : access (33)
  Subject: Compartments : none
            Permissions : suidgid
                  Class : 0
             Categories : none
            Access Mode : write
  Object: Level: 0  uid: root(0) gid:root(0) device:0, 235 inode:3412
           Compartments : none
                  Class : 0
             Categories : none
                   Mode : 100755
```

The following example shows the discretionary access violation record that is produced when the `-S` option is specified. When this option is used, the `Subject: Compartments`, `Object: Level`, and `Compartments` information is not printed in the body of the record, but in the record header. See Section 8.8.8.2, page 345, for more information on this option:

```
$ /etc/reduce -t disc -S
Oct 21 13:24:22 1993  Discretionary   jid:945  pid:79982
r_ids:[operator(9),operator(9)]  e_ids:[operator(9),operator(9)]
    S_Label: 0,none  O_Label: 0,none
Login uid: operator(9)

  Function: open (5)     Violation: Permission denied (13)
            System call : access (33)
  Subject: Permissions : none
                  Class : 0
             Categories : none
            Access Mode : write
          Object: 0  uid: root(0) gid:root(0) device:0, 235 inode:3412
           Compartments : none
                  Class : 0
             Categories : none
                   Mode : 40755
```

To show the full path name to the object, use the -p option, as shown in the following example. The SLG_PATH_TRACK parameter must be enabled to generate this information.

```
$ /etc/reduce -p -t disc

Apr  1 09:50:00 1991  Discretionary o_lvl: 0 s_lvl: 0 jid:167 pid:7026
  r_ids:[root(0),root(0)] e_ids:[root(0),root(0)]           **********
Login uid: root(0)

  Function: open (5)     Violation: Permission denied (13)
            System call : open (5)
  Subject: Compartments : none
            Permissions : suidgid
                  Class : 0
             Categories : none
            Access Mode : write
  Object: Level: 0  uid: root(0) gid:root(0) device:0, 235 inode:3412
               Pathname : /etc/utmp
           Compartments : none
                  Class : 0
             Categories : none
                   Mode : 100755
```

For the `shmsys`, `semsys`, and `msgsys` system calls, the `SLG_DISC_7` record format displays the header information and the following information:

```
           System call: syscall(number)  Violation: text (nnn)
Subject: Categories   : sub_intcat
Object:      Owner uid: user(ID)
             Owner gid: group(ID)
           Creator uid: creator(ID)
           Creator gid: group(ID)
           Access mode: mode
  Slot sequence number: seq_number
                   Key: key_number
```

The majority of these fields are self-explanatory, except for the following fields, which are defined as follows:

Field	Description
`Slot sequence number`	Indicates the sequence number of the slot for the IPC object. A slot is the entry in a kernel table for the IPC object.
`Key`	Indicates the user-selected, 64-bit identifier for the IPC object. For a private IPC object, the key is 0; in other cases, it is a hex number used in the `reduce` output.

The following example shows the record displayed for a `semsys` system call:

```
$ /etc/reduce -t disc

Jul 27 17:34:19 1994  Discretionary o_lvl:0 s_lvl:0 jid:34  pid:1804
r_ids:[vsx0(36056),vsxg0(31000)] e_ids:[root(0),vsxg0(31000)] ******
   Login uid: vsx0(36056)

       System call : semsys (53)   Violation: Permission denied (13)
  Subject: Categories : none
  Object:    Owner uid : 36056 (vsx0)
             Owner gid : 31000 (vsxg0)
           Creator uid : 36056 (vsx0)
           Creator gid : 31000 (vsxg0)
           Access Mode : read write
  Slot sequence number : 1
                   Key : 0
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.7 Discretionary access change record (SLG_DAC_CHNG)

The discretionary access change record is written for all successful and failed attempts to change the discretionary access control (DAC) attributes of a file. It also records changes to the access control list (ACL) and owner of a file. The SLG_DAC_CHNG record type is used to record this information.

On UNICOS 7.0 and earlier systems, this record type was called the SLG_SUID record, because only changes to setuid file attributes were being audited (although this record type was used to indicate changes to the file permissions mode bits also).

DAC changes are logged for the following system calls: getfacl(2), setfacl(2), rmfacl(2), chmod(2), fchmod(2), chown(2), and fchown(2).

The SLG_LOG_DAC parameter must be on to log this record type To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log discretionary access changes? selection in the UNICOS Installation and Configuration Menu System.

You can also use the spaudit -e dac command to enable the generation of this record or the spaudit -d dac command to disable the generation of this record.

In addition to the header information, the SLG_DAC_CHNG record with path tracking enabled contains the following information:

```
Function: func (nnn)     Violation: text (nnn)
          System call : syscall (nnn)
Subject: Compartments : sub_cmpts
          Permissions : sub_permits
                Class : sub_intcls
           Categories : sub_intcat
Object: Level: lvl  uid: user(ID)  gid: user(id)  device: maj, min
inode: INUM
             Pathname : /pathname
         Compartments : obj_comps
                Class : obj_intcls
           Categories : obj_intcat
                 Mode : obj_mode
```

The field definitions are the same as for for the discretionary access violation record. See Section 8.8.7.6, page 280, for a description of the format.

To display a DAC change record for a `fchmod`(2) system call with object path tracking enabled, use the `reduce` command, as shown in the following example. The `-S` option is used in this example, which means that the `Subject: Compartments` and `Compartments : obj_comps` fields are not printed in the body of the record, but in the record header. See Section 8.8.8.2, page 345, for more information on this option:

```
$ /etc/reduce -t dac -S -p
Oct 21 13:24:22 1993  DAC Change    jid:945  pid:79982
r_ids:[root(0),root(0)]  e_ids:[root(0),root(0)]
    S_Label: 0,none  O_Label: 0,none
    Login uid: root(0)

   Function: chown (16)     Violation: NONE (0)
              System call : chown (16)
    Subject: Permissions : none
                    Class : 0
               Categories : none
  Object: Level: 51  uid: root(0) gid: root(0)
                           device: 0, 5 inode: 555
                 Pathname : /etc/mnttab
                    Class : 0
               Categories : none
                     Mode : 40755
```

The following example shows the DAC change record for attaching an ACL to a file:

```
$ /etc/reduce -t dac -p
Jul 15 10:12:29 1993  DAC Change  o_lvl: 0 s_lvl: 0 jid:1214 pid:33078
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
    Login uid: operator(9)

   Function: setacl (87)     Violation: NONE (0)
             System call : setacl (87)
   Subject: Compartments : none
             Permissions : suidgid
                   Class : 0
              Categories : none
   Object: Level: 0  uid: operator(9)  gid: operator(9)
                            device: 0, 8  inode: 5008
                Pathname : cmd/myfile
            Compartments : none
                   Class : 0
              Categories : none
                    Mode : 100600
```

The following is an example of a DAC change record for a `chown`(2) system call:

```
$ /etc/reduce -t dac -p
Jul 15 10:59:13 1993  DAC Change   o_lvl: 0 s_lvl: 0 jid:1274 pid:34497
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
    Login uid: operator(9)

   Function: chown (16)     Violation: NONE (0)
             System call : chown (16)
   Subject: Compartments : none
             Permissions : suidgid
                   Class : 0
              Categories : none
   Object: Level: 0  uid: root(0) gid: root(0)
                            device: 0, 22 inode: 1784
                Pathname : /tmp/maAAAa34497
            Compartments : none
                  Class :   0
              Categories : none
                    Mode : 100000
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.8 Mandatory access record (SLG_MAND_7)

The mandatory access record is written for successful file system object accesses, for mandatory access violations that occur during access to a file system object or its path, and for mandatory access violations with process-to-process accesses. The SLG_MAND_7 record is used.

A SLG_MAND_7 record is written for the following system calls when a successful file system object access attempt is made or when an unsuccessful mandatory access violation occurs during access to a file system object or its path: access(2), acct(2), chacid(2), chdir(2), chmod(2), chown(2), chroot(2), creat(2), dacct(2), exec(2), fchmod(2), fchown(2), fcntl, fgetpal(2), fstat, getacl (2), getdents, jacct(2), join(2), lchown(2), lsecstat(2), lstat(2), mkdir(2), mknod(2), open(2), pathconf(2), quotactl, readlink(2), rename(2), restart(2), rmfacl(2), secstat(2), select(2), setacl (2), setdevs(2), setfcmp(2), setflvl(2), setpal(2), stat(2), statfs(2), symlink(2), and utime(2).

A SLG_MAND_7 record is automatically written for setflvl(2), setfcmp(2), and setfflg(2) system calls, regardless of success or failure.

The SLG_MAND_7 record for mandatory access failures for process-to-process accesses is written by the following system calls: acctid(2), chkpnt(2), cpselect(2), getlim(2), kill(2), killm(2), limit(2), nicem(2), ptyrecon(2), resume(2), setlim(2), and suspend(2).

The SLG_MAND_7 record for mandatory access failures is automatically written for the following system calls for mandatory access failures on accesses and for mandatory access failures or successes when removing one of the IPC objects: msgctl(2), msgget(2), msgrcv(2), msgsnd(2) semctl(2), semget(2), semop(2), shmat(2),shmctl(2), shmdt(2), shmget(2). For removal failures to be audited, the SLG_ALL_RM configuration parameter must be enabled. For the mandatory access control failures on accesses to be audited, the SLG_MANDV configuration parameter must be enabled. See Section 8.8.7.6, page 280, for a description of the format of these records.

The SLG_MANDV parameter must be on for mandatory access violations to be logged.

To set this parameter, use the `Configure system ->Multilevel security (MLS) configuration->Security log file configuration->Log mandatory access violations?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e mandv` command to enable the generation of this record. The `spaudit -d mandv` command disables the generation of this record.

The `SLG_ALL_VALID` parameter must be on for all successful file system object accesses to be logged. All of the object fields in the record may be 0, depending on what type of error is encountered. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all access attempts?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e all_valid` command to enable the generation of this record for successful accesses. The `spaudit -d all_valid` command disables the generation of this record.

The format of this record is the same as described previously for the discretionary access violation record. See Section 8.8.7.6, page 280, for a description of the format.

To display a `SLG_MAND_7` record, use the `reduce` command, as shown in the following example. The example shows the actual system call causing the violation is shown. This line is printed when the system call is different than the function. In this example, the `creat`(2) system call called the `open`(2) function, causing the violation. The `(8)` after the `creat` in the following example is the actual system call number:

```
$ /etc/reduce -p -t mand

Apr  1 09:52:42 1991  Mandatory   o_lvl: 0  s_lvl: 0  jid:17  pid:7476
  r_ids:[root(0),root(0)]      e_ids:[root(0),root(0)]      **********
Login uid: operator(9)

  Function: open (5)     Violation: Security category violation (319)
             System call : creat (8)
   Subject: Compartments : none
             Permissions : suidgid
                   Class : 0
              Categories : secadm sysadm
             Access Mode : write
  Object: Level: 0   uid: root(0) gid:sys(3) device:0, 225 inode:926
                Pathname : /etc/udb
            Compartments : none
                   Class : 0
              Categories : none
                    Mode : 100600
```

The following example shows the record produced for a process-to-process access violation. This example shows the use of the `-S` option. See Section 8.8.8.2, page 345, for more information on this option:

```
$ /etc/reduce -t mand -S

Nov 19 09:29:51 1993  Mandatory  jid:38  pid:1843
r_ids:[root(0),root(0)]      e_ids:[root(0),root(0)]
    S_Label: 4,comp24  O_Label: 7,none
    Login uid: root(0)

             System call : limit (115)   Violation: No such process (3)
    Subject: Categories : none
    Object:       Job ID : 50
              Process ID : 1810
             Process uid : 0 (root)
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.9 Login validation record (SLG_LOGN)

The SLG_LOGN record is written for all successful and unsuccessful login attempts. This includes all login attempts that are done interactively through NQS, ftp(1B), rlogin(1), dgdemon(8), and rshd(8). The cron(8) command also issues this record whenever a crontab(1) or at(1) job is initiated on behalf of a user. For sublogins, the audit record that is written is for the initial successful login only. This matches the user ID and job ID in the job termination record.

Except for sublogins, the login user ID and job ID that are written in this record remain unchanged throughout the life of a job on the system and can be used to track all activities of this job from initiation to completion. For sublogins, the audit record contains the initial user ID or the second login user ID. For sublogins, after the job initiation record is written, audit records that are generated during execution of the job may contain the second login user ID.

The entries are logged through the slgentry(2) system call.

If a login failure is because the user's name is not valid (that is, not in the UDB), the failure is logged as an invalid user ID. The text string for the invalid user name is not logged. This is done to ensure that if a user inadvertently types his or her password in the ID prompt, the clear text password is not written in the security log. The entries are logged through slgentry(2).

The SLG_JSTART parameter must be on to enable the generation of this record. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log start of job?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e jstart` command to enable the generation of this record. The `spaudit -d jstart` command disables the generation of this record.

The SLG_LOGN record type uses the SLG_LOG_USER record subtype to record clear text passwords for failed logins. If this record is enabled and the user is trapped (by having the usrtrap permission assigned in the user database (UDB) entry), and the user's login attempt fails, the password is recorded as part of the record. This means that passwords are sometimes recorded in the security log.

The SLG_USER parameter must be on to record the clear text password violations (assuming the SLG_JSTART parameter is also enabled). Again, the entered password is recorded only for a failed login attempt when the usrtrap permission is set for the user. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log name and password on login password fail?` selection in the UNICOS Installation and Configuration Menu System.

The `spaudit -e user` command enables the user password to be recorded for failed password attempts by a trapped user. The `spaudit -d user` disables this ability.

In addition to the header information, the `SLG_LOGN` record displays the following information:

```
Login: [ssss(n),ssss(n)] : Result  Host  tty line  Failures
```

The fields are defined as follows:

Field	Description
`Login: [ssss(n),ssss(n)]`	The user, group names, and IDs in the form of [*username(user ID),groupname(group ID)*].
`Result`	Indicates whether the login was successful (signified by `Okay`) or displays a violation message if the login was unsuccessful. The following list explains the login violation messages:

Message	Description
`Lastlog`	`login` was unable to update the user's `Last Login Time` entry in the UDB.
`Password`	The user entered an invalid password.
`Setusrv`	User's defined security label is invalid, outside the range of the system's label, or outside the range defined in the NAL.
`Shell exec`	User was not found in the `/etc/utmp` file.
`Locked`	Account is disabled in the UDB.

`Disabled`	Account is disabled because the `logfails` field in the UDB exceeds the limits defined by `MAXLOGS`.
`Dialup`	An invalid dialup password was used.
`Trusted subject`	User has authorized `system` or `daemon` category.
`Root console`	The user attempted to log in as `root` from a console other than the system console.
`Setjob failed`	Unable to set up login process with a new job ID and table entry.
`Attempted to login to localhost`	A login attempt was not allowed for `localhost`.

There are four different message formats:

Format	Description
Message `error`	*Message* describes the reason for the failure (as defined in the previous list).
Message `error (disabled)`	*Message* describes the reason for the failure, while `(disabled)` means that account is disabled because `MAXLOGS` was equaled or exceeded. See Section 8.5.5.11, page 222, for more information on how `MAXLOGS` works. The errors most likely to use this message format are the `Password` and `Disabled` messages. Until an account is disabled, only the message portion appears. For example, if the password is incorrectly entered, the `Password error` message is recorded.

The first time MAXLOGS is equaled, in addition to the *Message*, the (disabled) portion of the message also appears. For example, if an incorrect password is entered until MAXLOGS is equaled, the Password error (disabled) message is recorded in the log.

Any subsequent attempts to log into an account after the first time MAXLOGS is equaled is logged with the Disabled error (disabled) message. The account stays disabled until it is cleared.

Message error (disabled)	*Message* describes the reason for the failure, and (disabled) means the account is disabled because MAXLOGS was exceeded.

Host	Name of remote host
tty line	The tty line used for the login attempt or the source of the login attempt (for example, cron or rshd).
Failures	Number of login failures to date

To display a SLG_LOGN record, use the reduce command, as shown in the following examples. The first example shows a record produced when a user logs in correctly and has no previous login errors:

```
$ /etc/reduce -t logn

Apr 29 08:12:57 1991  Validation      o_lvl: 0  s_lvl: 0  jid:0 pid:994
  r_ids:[operator(9),operator(9)]  e_ids:[operator(9),operator(9)]
    **********


 Login to [operator(9),operator(9)] : Okay via ows131 on /dev/console
```

The following example shows the record produced when a user logs in correctly, but has a previous login failure:

```
$ /etc/reduce -t logn

Apr 29 08:14:21 1991  Validation     o_lvl: 0  s_lvl: 0  jid:0 pid:1067
  r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
*********
Login uid: operator(9)

 Login to [operator(9),operator(9)] : Okay   via juniper04
 on /dev/ttyp001 -- 1 previous failures
```

The following example shows the record produced when a user entered an incorrect password:

```
$ /etc/reduce -t logn

Apr 29 08:14:15 1991  Validation     o_lvl: 0  s_lvl: 0  jid:0 pid:1067
  r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
*********
Login uid: operator(9)

  Login to [operator(9),operator(9)] : Password error  via juniper04
on /dev/ttyp001 -- 0 previous failures
```

The following example shows the record produced when a user has exceeded the maximum number of login attempts (defined by `MAXLOGS`):

```
$ /etc/reduce -t logn

Apr 29 09:07:55 1991  Validation     o_lvl: 0  s_lvl: 0  jid:0 pid:2456
  r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
  **********
Login uid: operator(9)

  Login to [operator(9),operator(9)] : Disabled error (disabled)
via fir28 on /dev/ttyp066 -- 5 previous failures
```

The following example shows the record produced when an administrator has set the `disabled` field in the UDB for a user. This record is produced when the disabled user attempts to log in:

```
$ /etc/reduce -t logn

Apr 29 09:10:28 1991  Validation     o_lvl: 0  s_lvl: 0  jid:0 pid:2907
  r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
  **********
Login uid: operator(9)

  Login to [operator(9),operator(9)] : Locked error  via fir28
on /dev/ttyp066 -- 0 previous failures
```

The following example shows the record produced for a trapped user with a password failure and SLG_USER is enabled:

```
$ /etc/reduce -t logn
Jul 15 10:45:47 1993  Validation   o_lvl: 0 s_lvl: 0 jid:0 pid:34302
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
    Login uid: operator(9)

  Login to [operator(9),operator(9)] : Password error  via cherry21
on /dev/ttyp012 - - 0 previous failures
name: operator, password: badpass
```

The following example shows the record produced for a user who logged in through the remote shell:

```
$ /etc/reduce -t logn

Oct 18 10:55:43 1993  Validation o_lvl:0  s_lvl:0  jid:3389  pid:65592
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**** ******
    Login uid: operator(9)

  Login to [operator(9),operator(9)] : Okay   via pecan10 on   rshd
```

The following example shows the record produced for a user who initiated a job through cron(8):

```
$ /etc/reduce -t logn

Oct 18 11:00:01 1993  Validation o_lvl:0  s_lvl:0  jid:3393  pid:66918
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
    Login uid: operator(9)

  Login to [operator(9),operator(9)] : Okay via cron on
```

The following example shows the user to produce the record header with the subject and object label information. See Section 8.8.8.2, page 345, for more information on this option:

```
$ /etc/reduce -t logn -S

Oct 21 13:24:17 1993  Validation  jid:1094  pid:79926
r_ids:[operator(9),operator(9)]  e_ids:[operator(9),operator(9)
    S_Label: 0,none  O_Label: 0,none
    Login uid: operator(9)

  Login to [operator(9),operator(9)] : Okay via cherry21 on /dev/ttyp043
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.10 Tape activity record (`SLG_TAPE`)

All records generated by `tpdaemon`(8) are logged using the `SLG_TAPE` record. The entries are logged through `slgentry`(2).

The following two additional events generate audit records by `tpdaemon`:

- Mandatory access control (MAC) failures on requests to reserve a device.
- Any tape MAC mediation performed by `tpdaemon`; this event is never recorded on a Cray ML-Safe system configuration, as all tape mediation is performed by Cray/REELlibrarian.

The `SLG_TAPE` record is generated by the following tape commands: `rls`(1), `rsv`(1), `tpapm`(8), `tpcatalog`(1), `tpdev`(8), `tpdstop`(8), `tpfrls`(8), `tpgstat`(8), `tpmls`(8), `tpmnt`(1), `tprst`(1), `tpscr`(8), `tpset`(8), `tpstat`(1), and `tpu`(8).

The SLG_TRUST record is generated by the tpconfig(8) command. See Section 8.8.7.24, page 332, for more information on this record type.

The SLG_LOG_TAPE parameter must be on for this record to be generated. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log tape activity? selection in the UNICOS Installation and Configuration Menu System.

You can also use the spaudit -e tapes command to enable the generation of this record or the spaudit -d tapes command to disable the generation of this record.

The message file field in the SLG_TAPE record is used only by the rsv(1) command. The external VSN field is used only on commands that take a VSN as a parameter. The violation field is the tpdaemon error; otherwise, for commands, it is the return code from tpdaemon.

To display this record, use the reduce command as shown in the following example:

```
$ /etc/reduce -t tape
Nov  9 16:45:17 1993  Tape I/O  o_lvl: 0 s_lvl: 0 jid:1158  pid:33559
r_ids:[operator(9),operator(9)]  e_ids:[root(0),root(0)]  **********
    Login uid: operator(9)

             NQS batch job ID:
                    Operation: Tape Mount
                    violation: 0
         permitted privileges: PRIV_NULL
               tape file name: AAAA33531
                 request pipe: /usr/spool/tape/daemon.request
                   reply pipe: /usr/spool/tape/1158tpmn33559
                 message file:
                 external VSN: 001815
```

The following example shows the record generated by the tpstat(1) command:

```
$ /etc/reduce -t tape
Dec  3 11:12:43 1993  Tape I/O   o_lvl: 0  s_lvl: 0  jid:514  pid:32969
r_ids:[operator(9),operator(9)]   e_ids:[root(0),root(0)]   **********
    Login uid: operator(9)

             NQS batch job ID:
                    Operation: Tape Status Request
                    violation: 0
         permitted privileges: PRIV_NULL
               tape file name:
                 request pipe: /usr/spool/tape/daemon.request
                   reply pipe: /usr/spool/tape/514tpst32969
                 message file:
```

The following example shows the record generated by the tape daemon for a `rsv`(1) command:

```
$ /etc/reduce -t tape
Dec  3 11:13:26 1993  Tape I/O   o_lvl: 0  s_lvl:54  jid:9  pid:32987
r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]           **********
    Login uid: root(0)

             NQS batch job ID:
                    Operation: User Reserve Status Request
                    violation: 0
         permitted privileges: PRIV_NULL
               tape file name:
                 request pipe: /usr/spool/tape/daemon.request
                   reply pipe: /usr/spool/tape/9gsta32987
                 message file:
```

The following example shows the record generated by the `rsv`(1) command:

```
$ /etc/reduce -t tape
Dec  3 11:13:30 1993  Tape I/O  o_lvl: 0  s_lvl: 0  jid:514  pid:32990
r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
          **********
    Login uid: operator(9)

             NQS batch job ID:
                    Operation: Reservation Request
                    violation: 0
         permitted privileges: PRIV_DAC_OVERRIDE,PRIV_MAC_READ,
                               PRIV_MAC_WRITE
               tape file name:
                 request pipe: /tmp.mld/jtmp/jtmp.000514a/TAPE_REQ_514
                   reply pipe: /usr/spool/tape/514rsv32990
                 message file: /drizzle/operator/tape.msg
```

The following example shows the record generated by the `tpmnt`(1) command:

```
$ /etc/reduce -t tape
Dec  3 11:14:01 1993  Tape I/O o_lvl: 0  s_lvl: 0  jid:514  pid:32998
r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
           **********
    Login uid: operator(9)

             NQS batch job ID:
                    Operation: Tape Mount
                    violation: 0
         permitted privileges: PRIV_NULL
               tape file name: PSU
                 request pipe: /usr/spool/tape/daemon.request
                   reply pipe: /usr/spool/tape/514tpmn32998
                 message file:
                 external VSN: 002895
```

The following example shows the record generated by the `rls`(1) command:

```
$ /etc/reduce -t tape
Dec  3 11:14:06 1993  Tape I/O  o _lvl: 0  s_lvl: 0  jid:514  pid:33000
r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
           **********
   Login uid: operator(9)

             NQS batch job ID:
                    Operation: Free Device Group
                    violation: 0
         permitted privileges: PRIV_NULL
               tape file name:
                 request pipe: /usr/spool/tape/daemon.request
                   reply pipe: /usr/spool/tape/514rls33000
                 message file:
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.11 End-of-job record (SLG_EOJ)

The SLG_EOJ record is written when a user's session exits the system, regardless of how the session originates (that is, whether the session is initiated through batch or interactive means, cron(8), rlogin(1), rshd(8), or through the setsid(2) system call). This record can be correlated with a SLG_LOGN record through the login user ID field and the job ID field. Except for sublogins, these fields are the same in both the SLG_LOGN and SLG_EOJ records. For sublogins, the login user ID in a SLG_EOJ record is the initial login user ID; in this case, the job ID should be used to track a sublogin SLG_EOJ record.

The SLG_JEND parameter must be on for this record to be generated. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log end of job?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e jend` command to enable the generation of this record or the `spaudit -d jend` command to disable the generation of this record.

Only the header information is produced for this record, as shown in the following example:

```
$ /etc/reduce -t eoj

Apr  1 09:56:46 1991  End of Job   jid:188  uid:operator(9) **********
    Login uid: operator(9)



Apr  1 09:57:05 1991  End of Job   jid:58  uid:operator(9) **********
    Login uid: operator(9)



Apr  1 09:57:05 1991  End of Job   jid:59  uid:root(0) **********
    Login uid: root(0)
```

See Section 8.8.7.1, page 274, for a description of the header information.

8.8.7.12 Change directory record (SLG_CHDIR)

The SLG_CHDIR record is produced each time a chdir(2) system call is made (that is, whenever users change their directories).

The SLG_LOG_CHDIR parameter must be on for this record to be generated. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log chdir requests? selection in the UNICOS Installation and Configuration Menu System.

In addition, the SLG_PATH_TRACK parameter must be on for this record to be generated. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Track all path names on accesses? selection in the UNICOS Installation and Configuration Menu System.

You can also use the spaudit -e chdir,object_path command to enable the generation of this record. The spaudit -d chdir,object_path command disables the generation of this record.

In addition to the header information, the SLG_CHDIR record displays the following information:

```
Relative directory path: rel_path
Absolute directory path: abs_path
```

The fields are defined as follows:

Field	Description
`rel_path`	The relative path name to the object
`abs_path`	The absolute path name to the object (usually starts with `/`). It is possible to get a path name that does not start with `/`. For example, if you are using `reduce` to examine a new security log file and your original login (and the `cd $HOME`) record was in an archived log file, then `reduce` would be unable to determine where you started. Instead, it creates as much of the path name as possible.

To display a `SLG_CHDIR` record, use the `reduce` command, as shown in the following example, which shows the `-S` option used to produce the record header with the subject and object label information: See Section 8.8.8.2, page 345, for more information on this option.

```
$ /etc/reduce -S -t chdir
Oct 21 13:24:18 1993  Change Directory jid:1094 pid:79963
r_ids:[operator(9),operator(9)]  e_ids:[operator(9),operator(9)]
    S_Label: 0,none  O_Label: 0,none
    Login uid: operator(9)

    Relative directory path: ./libc
    Absolute directory path: /ulib/usr/src/lib/libc



$ /etc/reduce -t chdir -s 04010930
Apr  1 09:59:10 1991  Change Directory o_lvl:0 s_lvl:0 jid:192 pid:9232
    r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]        **********
Login uid: root(0)

    Relative directory path: /usr/spool/nqs/private/root
    Absolute directory path: /usr/spool/nqs/private/root
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.13 Security-related system call record (SLG_SECSYS)

The SLG_SECSYS record is written whenever one of the following security-related system calls is executed: fsetpal(2), setjob(2), setpal(2), setsid(2), setucat(2), setucmp(2), setulvl(2), and setusrv(2).

It is also written if an authorization error or system error occurs when a process issues the slgentry(2) system call. The slgentry system call is used by the Cray ML-Safe process to write the record to the security log.

The SLG_LOG_SECSYS parameter must be on for this record to be generated. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log security system calls? selection in the UNICOS Installation and Configuration Menu System.

You can also use the spaudit -e secsys command to enable the generation of this record. The spaudit -d secsys command disables the generation of this record.

There are three formats for this record. The first format is used when the setucat(2) and setusrv(2) system calls are executed. This format is also used for an authorization error or system error occurs when a process issues the slgentry(2) system call. For the first format, in addition to the header information, the SLG_SECSYS record displays the information shown in the following example.

```
  Function: func (nn)        Violation: text (nnn)
 Subject: Active Compartments : sub_active_comps
          Valid Compartments : sub_val_comps
                 Permissions : sub_permits
                       Class : sub_intcls
           Active Categories : sub_act_cat
            Valid Categories : sub_val_cat
```

The fields are defined as follows:

Field	Description
Function: func (nn)	The function name and its system call.

`Violation:` `text (nnn)`

The error message associated with this violation followed by the error number. If no violation occurred, the value `NONE(0)` is shown. See Section 8.8.10, page 354, for more information.

`sub_act_comps`

The subject's active security compartments.

`sub_val_comps`

The subject's authorized security compartments.

`sub_permits`

The subject's authorized permissions (as defined in the UDB).

`sub_intcls`

The subject's active integrity class. This value is no longer used.

`sub_act_cat`

The subject's active category.

`sub_val_cat`

The subject's authorized categories.

To display the first format of a `SLG_SECSYS` record, use the `reduce` command, as shown in the following example:

```
$ /etc/reduce -t secsys

Apr  1 10:02:35 1991 Security Syscall o_lvl:0 s_lvl:0 jid:212 pid:11855
 r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]         **********
Login uid: root(0)

   Function: setusrv (89)     Violation: NONE (0)
  Subject: Active Compartments : none
            Valid Compartments : none
                   Permissions : suidgid
                         Class : 0
             Active Categories : none
              Valid Categories : secadm


Apr  1 10:02:37 1991 Security Syscall o_lvl:0 s_lvl:0 jid:212 pid:11855
  r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]        **********
Login uid: root(0)

   Function: setucat (154)     Violation: NONE (0)
  Subject: Active Compartments : none
            Valid Compartments : none
                   Permissions : suidgid
                         Class : 0
             Active Categories : none
              Valid Categories : system



Apr  1 10:02:39 1991 Security Syscall o_lvl:0 s_lvl:0 jid:212 pid:11855
  r_ids:[root(0),operator(9)]   e_ids:[root(0),operator(9)]  ********
Login uid: operator(9)

   Function: setusrv (89)     Violation: NONE (0)
  Subject: Active Compartments : none
            Valid Compartments : none
                   Permissions : suidgid
                         Class : 0
             Active Categories : none
              Valid Categories : secadm
```

The second format of the SLG_SECSYS record is used by the setsid(2) and setjob(2) system calls to audit the creation of a new job or session ID from within an existing session. The purpose of this record is to document the changing of the job or session ID so that all actions taken by the new job or session can be traced to the original session and user ID.

In addition to the header information, this form of the SLG_SECSYS record displays the following information:

```
   Function: func: (nn)     Violation: text (nnn)
  Subject: Active Compartments : sub_active_comps
            Valid Compartments : sub_val_comps
                   Permissions : sub_permits
                         Class : sub_TFM_class
             Active Categories : sub_act_cat
              Valid Categories : sub_val_cat
                    New job id : new_job_id
                   New job uid : new_uid (nnnn)
```

To display the second format of a SLG_SECSYS record, use the reduce command, as shown in the following example:

```
$ /etc/reduce -t secsys

Oct 18 14:47:33 1993 Security Syscall o_lvl:0 s_lvl:0 jid:4455 pid:65926
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
    Login uid: operator(9)

   Function: setsid (117)     Violation: NONE (0)
  Subject: Active Compartments : none
            Valid Compartments : none
                   Permissions : suidgid
                         Class : 0
             Active Categories : none
              Valid Categories : secadm sysadm
                    New job id : 4456
                   New job uid : dnn (1346)
```

A third form of the SLG_SECSYS record is written when the setpal(2) system call is executed. The following example shows this type of record. The subject and object security labels are supplied in the following format: level, compart1 compart2,.... The first record is generated by a setprivs(8) command and no privilege assignment lists (PALs) entries were used. The second record was generated by a `setpal`(1) command and three PAL entries were set.

This record type can contain up to eight PAL entires, which is enough for all software released by Cray Research. If a record generated by `setpal`(2) contains more than eight PAL entries, the additional records are written until all PAL entries have been logged:

```
$ /etc/reduce -t secsys

Jul 15 11:15:12 1993 Security Syscall o_lvl:0 s_lvl:0 jid:1274 pid:34930
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
  Login uid: operator(9)

     Function: setpal (220)     Violation: NONE (0)
     Subject: active categories: secadm
                         Label: 0, none
     Object:             Label: 0, none
                           uid: operator(9)  gid: operator(9)
                        device: 0, 8  inode: 5244
                      Pathname: mybin
            allowed privileges: PRIV_NULL
             forced privileges: PRIV_DAC_OVERRIDE,PRIV_MAC_READ,
                                PRIV_MAC_WRITE
       set effective privileges: PRIV_DAC_OVERRIDE,
                                 PRIV_MAC_READ,PRIV_MAC_WRITE
             total PAL entries: 0
            PAL version number: 0
                   PAL entries: 0
```

```
Jul 15 11:15:38 1993 Security Syscall o_lvl:0 s_lvl:0 jid:1274 pid:34933
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
  Login uid: operator(9)

    Function: setpal (220)     Violation: NONE (0)
    Subject: active categories: secadm
                         Label: 0, none
    Object:              Label: 0, none
                           uid: operator(9)  gid: operator(9)
                        device: 0, 8  inode: 5244
                      Pathname: mybin
            allowed privileges: PRIV_NULL
             forced privileges: PRIV_DAC_OVERRIDE,PRIV_MAC_READ,
                                PRIV_MAC_WRITE
      set effective privileges: PRIV_DAC_OVERRIDE,PRIV_MAC_READ,
                                PRIV_MAC_WRITE
             total PAL entries: 3
            PAL version number: 0
                   PAL entries: 3

                 privilege set: PRIV_DAC_OVERRIDE,PRIV_MAC_READ,
                                PRIV_MAC_WRITE
                privilege text: exec
           privileged category: secadm

                 privilege set: PRIV_DAC_OVERRIDE,PRIV_MAC_READ,
                                PRIV_MAC_WRITE
                privilege text: exec
           privileged category: sysadm

                 privilege set: PRIV_DAC_OVERRIDE,PRIV_MAC_READ,
                                PRIV_MAC_WRITE
                privilege text: exec
           privileged category: system
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.14 NAMI function record (SLG_NAMI)

This record is produced whenever a file's path is searched by any of the following system calls: `link`(2), `mkdir`(2), `rmdir`(2), and `unlink`(2). If logging of this record is enabled, it is written in addition to any other type of discretionary or mandatory access records that are written.

The format is the same as described for the discretionary access violation record. See Section 8.8.7.6, page 280, for a description of the format.

The `SLG_MKDIR`, `SLG_RMDIRV`, `SLG_LINKV`, `SLG_REMOVEV`, `SLG_ALL_RM`, and/or `SLG_ALL_NAMI` parameters must be on for this record to be generated.

The `SLG_MKDIRV` parameter allows the logging of all `mkdir`(2) system call violations.

To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all directory make (mkdir) violations?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e mkdirv` command to enable the generation of this record or the `spaudit -d mkdirv` command to disable the generation of this record.

The `SLG_RMDIRV` parameter allows the logging of all `rmdir`(2) system call violations.

To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all directory remove (rmdir) violations?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e rmdirv` command to enable the generation of this record or the `spaudit -d rmdirv` command to disable the generation of this record.

The `SLG_LINKV` parameter allows the logging of all link violations, which includes mandatory access violation within link. This violation is also logged if the `SLG_MANDV` or `SLG_ALL_NAMI` configuration parameters are enabled.

To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all link (ln) violations?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e linkv` command to enable the generation of this record or the `spaudit -d linkv` command to disable the generation of this record.

All of the object fields in the record may be 0, depending on what type of error is encountered and which routine (`link` or `nami`) generated the record.

The `SLG_REMOVEV` parameter allows the logging of all file removal violations for the `unlink`(2) system call.

To set this parameter, use the `Configure system ->Multilevel security (MLS) configuration->Security log file configuration->Log all remove violations?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e removev` command to enable the generation of this record or the `spaudit -d removev` command to disable the generation of this record.

The `SLG_ALL_RM` parameter allows the logging of all remove (`unlink`(2)) requests regardless of the success or failure of the action.

To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all remove (rm) requests?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e all_rm` command to enable the generation of this record or the `spaudit -d all_rm` command to disable the generation of this record.

The `SLG_ALL_NAMI` parameter allows the logging of all `mkdir`(2), `rmdir`(2), `link`(2), and `unlink`(2) system calls regardless of success or failure of the call.

To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all mkdir, rmdir, link, and rm calls?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e all_nami` command to enable the generation of this record or the `spaudit -d all_nami` command to disable the generation of this record.

All of the object fields in the record may be 0, depending on what type of error is encountered.

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t nami -s 04011000 -p

Apr  1 10:04:24 1991  File Control   o_lvl:0 s_lvl:0 jid:102 pid:7496
  r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]       **********
Login uid: operator(9)

   Function: link (9)     Violation: File exists (17)
   Subject: Compartments : none
             Permissions : suidgid
                   Class : 0
              Categories : none
   Object: Level: 0  uid: root(0) gid:root(0) device:0, 192 inode:689
                Pathname : /fstest1/ram1/bo125
            Compartments : none
                   Class : 0
              Categories : none
                    Mode : 100644
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.15 setuid system call record (`SLG_SETUID`)

This record is produced whenever a `setuid`(2) or `setreuid`(2) system call is executed.

The `SLG_SUID_RQ` parameter must be on for this record to be generated. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all setuid requests?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e setuid` command to enable the generation of this record or the `spaudit -d setuid` command to disable the generation of this record.

In addition to the header information, the `SLG_SETUID` record displays the following information:

```
Setuid call from real_name (RUID) to eff_name (EUID) was text
```

The fields are defined as follows:

Field	Description
`real_name`	The real user login name
`RUID`	The real user ID
`eff_name`	The effective user login name
`EUID`	The effective user ID
`text`	Indicates if call was successful (`successful` or `NOT successful`)

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t setuid -s 04010800 -p

Apr  1 10:27:50 1991  Setuid Syscall o_lvl:0  s_lvl:0 jid:310 pid:21296
  r_ids:[operator(9),operator(9)]  e_ids:[operator(9),operator(9)]
**********
Login uid: operator(9)

   Setuid call from operator (9) to root (0) was NOT successful


Apr  1 10:27:58 1991  Setuid Syscall o_lvl:0  s_lvl:0 jid:312 pid:21343
  r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
Login uid: operator(9)

   Setuid call from operator (9) to operator (9) was successful


Apr  1 10:28:00 1991  Setuid Syscall o_lvl:0  s_lvl:0 jid:313 pid:21347
 r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]           **********
Login uid: root(0)

   Setuid call from root (0) to root (0) was successful

Apr  1 10:28:12 1991  Setuid Syscall o_lvl:0  s_lvl:0  jid:85 pid:9464
  r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
Login uid: operator(9)

   Setuid call from root (0) to operator (9) was successful
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.16 su attempt record (SLG_SU)

This record is produced for all successful or unsuccessful attempts to execute the su command, essentially producing the same information found in sulog.

The SLG_SULOG parameter must be on to generate this record. To set parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all su attempts? selection in the UNICOS Installation and Configuration Menu System.

You can also use the spaudit -e sulog command to enable the generation of this record or the spaudit -d sulog command to disable the generation of this record.

In addition to the header information, the SLG_SULOG record displays the following information:

```
Setuid call from real_name (RUID) to eff_name (EUID) was text
```

The fields are defined as follows:

Field	Description
real_name	The real user login name
RUID	The real user ID
eff_name	The effective user login name
EUID	The effective user ID
text	Indicates if call was successful (successful or NOT successful)

To display this record, use the reduce command as shown in the following example:

```
$ /etc/reduce -t sulog

Dec 11 05:10:07 1991  Sulog       o_lvl: 0  s_lvl: 0  jid:11  pid:3072
    r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]       **********
    Login uid: root(0)

   Setuid call from root (0) to adm (4) was successful
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.17 Networks security violations record (SLG_IPNET)

This record is produced for all network errors. The entries are logged through the slgentry(2) system call.

This record is generated only if the SLG_LOG_IPNET configuration parameter is enabled. To set parameter, use the Configure system->Multilevel `security (MLS) configuration->Security log file configuration->Log IP layer activity?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e ipnet` command to enable the generation of this record or the `spaudit -d ipnet` command to disable the generation of this record.

In addition to the header information, the `SLG_IPNET` record displays the following information:

The fields are defined as follows:

```
Function: requested_function     Violation: text (nnn)

      IP Security Option : type
       Network Interface : number
           Network Route : number

Subject:            Host : host
                   Level : sub_level
            Compartments : sub_comparts
Object:            Level : obj_level
            Compartments : obj_comparts
```

Field	Description
`Function:`	`requested_function`
	Defines one of the following functions: Set IP Security Option (1) or Get IP Security Option (2). A record with a requested function of 1 is generated when there is an error in the `ip_output` () routine. A record with a requested function of 2 is generated when there is an error on the IP input in the transport layer routines. In the latter case, the security header information is not relevant because the error occurred in an interrupt handler.
`Violation:`	`text (nnn)`
	Violation description and error number. See Section 8.8.10, page 354. for more information.

`IP Security Option :  type`

Defines one of the following IP security option types: None (0), BSO (1), or CIPSO (2)

`Network Interface :  number`

Reserved for future use

`Network Route :  number`

Reserved for future use

`Subject:  Host :  host`

Subject's host machine. If the remote host is not in `/etc/hosts`, then the Internet address is displayed. The `Subject` field refers to the local process or the remote host, depending on whether the datagram is outgoing or incoming, respectively.

`Level :  sub_level`

Subject's active security level

`Compartments :  sub_comparts`

Subject's active compartments

`Object:  Level :  obj_level`

Security level of incoming datagram

`Compartments :  obj_comparts`

Active compartments of the incoming datagram

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t netip

Dec 11 15:34:24 1991 Network IP Layer o_lvl: 0 s_lvl:16 jid:115 pid:5787
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
  **********
    Login uid: operator(9)

  Function: Set IP Security Option (1)   Violation: Host not authorized
  in NAL (330)

      IP Security Option : NONE (0)
       Network Interface : 0
           Network Route : 0

   Subject:         Host : pecan01.cray.com
                   Level : 16
            Compartments : none
   Object:         Level : 0
            Compartments : none
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.18 Cray NFS request record (SLG_NFS)

This record is produced for all Cray-to-Cray NFS requests, regardless of success or failure of the request.

The SLG_NFS parameter must be on to generate this record. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all Cray-to-Cray NFS requests` selection in the UNICOS installation and configuration menu system.

You can also use the `spaudit -e nfs` command to enable the generation of this record or the `spaudit -d nfs` command to disable the generation of this record.

In addition to the header information, the SLG_NFS record displays the following information:

```
   Function: requested_function     Violation: text (nnn)

 Remote Server (Client): address
```

```
     Session ID   : sid
 Transaction ID   : xid
  Device Number   : dev
   Inode Number   : inode
```

The fields are defined as follows:

Field	Description
`Function:  requested_function`	
	Cray NFS (CNFS) procedure number, which indicates what CNFS operation is returned
`Violation:  text (nnn)`	
	Violation description and error number. See Section 8.8.10, page 354. for more information.
`Remote Server (Client) :  address`	
	The hostname of the remote CNFS server or client, respectively. If the hostname is not known, the Internet address is shown.
`Session ID :  sid`	
	The session identification number of the client process that made the request
`Transaction ID :  xid`	
	The RPC packet transaction identification generated by the CNFS client
`Device Number :  dev`	
	The major or minor device number in the server's file systems of the file being accessed
`Inode Number :  inode`	
	The inode number in the server's file system of the file being accessed

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t nfs

Sept 10 15:17:32 1992 NFS     o_lvl: 0 s_lvl: 0 jid:629 pid:44007
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
  **********
    Login uid: operator(9)

   Function: lookup (4)     Violation: NONE (0)

           Remote Server : uss.cray.com
              Session ID : 629
          Transaction ID : 324f0b0bd17d44
           Device Number : 8767
            Inode Number : 4
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.19 File transfer record (SLG_FXFR)

This record is produced for all successful and failed `ftp`(1B) attempts. These entries are logged through the `slgentry`(2) system call.

The `SLG_FILEXFR` parameter must be on to generate this record. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all file transfers?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e filexfr` command to enable the generation of this record or the `spaudit -d filexfr` command to disable the generation of this record.

In addition to the header information, the `SLG_FXFR` record displays the following information:

```
Type_of File Transfer
Remote Host: ip_address  Remote Port: host_port  Local Port: host_port

Function: func (nn)   Violation: text (nnn)
Subject: Compartments : sub_comps
          Permissions : sub_permits
                Class : sub_intcls
           Categories : sub_intcat
          Access Mode : access_mode
Object: Level:lvl  uid: user(ID)  gid: group(ID)  device: maj,min
inode: INUM
             Pathname : file
         Compartments : obj_comps
                Class : obj_intcls
           Categories : obj_intcat
                 Mode : obj_mode
```

The fields are defined as follows:

Field	Description
`Type_of File Transfer`	Defines if the file is to be imported or exported by indicating `Incoming File Transfer` or `Outgoing File Transfer`.
`Remote Host:  ip_address`	The host name (or internet address) of the remote host.
`Remote Port:  host_port`	Remote host port number that is used for the transfer.
`Local Port:  host_port`	The Cray port number that is used for the transfer. If this port is registered in `/etc/services`, then the network service text is displayed in parentheses after the port number.

The rest of this record is formatted as described previously for the discretionary access violation record. See Section 8.8.7.6, page 280, for a description of the format.

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -p -t xfer
Dec 10 20:00:31 1992 File Transfer o_lvl:0 s_lvl:0 jid:5058  pid:82087
    r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]       **********
Login uid: operator(9)

Incoming File Transfer
Remote Host: fox         Remote Port: 1973  Local Port: 21 (ftp)

  Function: open (5)     Violation: Security category violation (319)
  Subject: Compartments  : none
             Permissions : suidgid
                   Class : 0
              Categories : none
             Access Mode : write
  Object: Level: 0   uid: root(3) gid:sys (3) device:0, 225 inode:926
                Pathname : /sn422/os/jnn/text
            Compartments : none
                   Class : 0
              Categories : none
                    Mode : 100600
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.20 Network configuration change record (SLG_NETCF)

This record is written when changes are made to the workstation access list (WAL), network access list (NAL), interface, and CIPSO map. These entries are logged through the `slgentry`(2) system call.

The `SLG_CF_NET` parameter must be on for this record to be generated. To set this parameter, use the **`Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all network configuration changes?`** selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e netcf` command to enable the generation of this record or The `spaudit -d netcf` command to disable the generation of this record.

The `SLG_NETCF` record uses the following five record subtypes:

- `SLG_NET_NAL` (records changes to the network access list (NAL))
- `SLG_NET_WAL` (records changes to the workstation access list (WAL))

- `SLG_NET_INTF` (records changes to the interface)
- `SLG_NET_MAP` (records changes to the CIPSO map)
- `SLG_NET_ROUTE` (records changes to the route)

In addition to the header information, the `SLG_NET_NAL` record displays the following fields, which are defined as follows:

Field	Description
`Function`	Can be `add` or `delete`, depending on whether the NAL entry was added or deleted, respectively.
`Address`	The host or network address for the entry
`Netmask`	The netmask for network entries
`Class`	The class value. This value is no longer used.
`Min level`	The minimum security level
`Min comparts`	The minimum compartment set
`Max level`	The maximum security level
`Access modes`	The access modes
`IP security option`	One of the following security options: Domain of interpretation for CIPSO (`DOI`); the required protection authority on input flags for BSO (`Authority in`); the required protection authority on output for BSO (`Authority out`)

In addition to the header information, the `SLG_NET_WAL` record displays the following fields, which are defined as follows:

Field	Description
`Function`	Can be `add` or `delete`, depending on whether the WAL entry was added or deleted, respectively.
`Address`	The host or network address for the entry
`Netmask`	The netmask for network entries
`uid.gid =3D services`	For each WAL permission; `uid` is the numeric user ID. `gid` is the numeric group ID or a * for

wildcard. services is the mask of allowed services.

In addition to the header information, the SLG_NET_INTF record displays the following fields, which are defined as follows:

Field	Description
Name	The interface name
Address	The interface address
Netmask	The interface netmask
Flags	The interface flags
Min level	The minimum security level
Min comparts	The minimum compartment set
Max level	The maximum security level
Max comparts	The maximum compartment set
Authority	The allowed BSO protection authority flags

In addition to the header information, the SLG_NET_MAP record displays the following fields, which are defined as follows:

Field	Description
Map ID	The numeric map ID
level hostlevel =3D netlevel	For each mapped level; hostlevel is the numeric host's level. netlevel is the numeric network's level.
compartment hostcmp =3D netcmp	For each mapped compartment; hostcmp is the host's compartment values. netcmp is the network's compartment values.

In addition to the header information, the SLG_NET_ROUTE record displays the following fields, which are defined as follows:

Field	Description
`Function`	Can be `add`, `change` or `delete`, depending on whether the route was added, changed, or deleted, respectively.
`Destination`	The destination of the route
`Netmask`	The route netmask

8.8.7.21 Audit criteria change record (`SLG_AUDIT`)

This record is produced whenever a change is made to change the configuration of the security logging options. The security logging configuration is changed when a security administrator uses the `spaudit -e` or `spaudit -d` to enable or disable the security logging options, respectively.

The `SLG_LOG_AUDIT` parameter must be on to generate this record. To set this parameter, use the `Configure system -> Multilevel security (MLS) configuration ->Security log file configuration->Log audit criteria changes?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e audit` command to enable the generation of this record or the `spaudit -d audit` command to disable the generation of this record.

In addition to the header information, the `SLG_AUDIT` record displays the following information. The output of this record is shows the state of all the auditing options following any changes that have been made:

```
$ /etc/reduce -t audit
Jul 15 11:09:54 1993  Audit Change o_lvl:0 s_lvl:0 jid:1274 pid:34814
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
 **********
    Login uid: operator(9)

             security auditing state: ON
                   all nami requests: OFF
                     all rm requests: OFF
                  all valid accesses: OFF
              audit criteria changes: ON
                      chdir requests: ON
             config changes (UNICOS): ON
                 Cray Reel Librarian: OFF
                         DAC changes: ON
            discretionary violations: ON
              file transfer requests: ON
                          I/O errors: OFF
              IPnet layer activities: ON
                             job end: ON
                      job initiation: ON
                     link violations: ON
                mandatory violations: ON
                    mkdir violations: ON
              network config changes: ON
                  network violations: ON
                        NFS activity: OFF
                        NQS activity: ON
                  NQS config changes: ON
                object path tracking: OFF
                    operator actions: ON
                       privilege use: OFF
                   remove violations: ON
                    rmdir violations: ON
                   security syscalls: ON
                     setuid requests: ON
                     system shutdown: ON
                      system startup: ON
                  system time change: ON
                         su requests: ON
                       tape activity: OFF
            trusted process activity: ON
         user password on login fail: OFF
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.22 NQS configuration change record (`SLG_NQSCF`)

This record is produced whenever a change is made to the configuration of NQS by using the `qmgr`(8) command. These entries are logged through the `slgentry`(2) system call.

The `SLG_CF_NQSCF` parameter must be on to generate this record. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log NQS configuration changes?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e nqscf` command to enable the generation of this record or the `spaudit -d nqscf` command to disable the generation of this record.

In addition to the header information, the `SLG_NQSCF` record displays the following information:

```
              Function: func
             Violation: type_violation
   Job sequence number: seqno
           Origin host: orighost
           Client host: chost
        Origin user id: origuid
        Target user id: tuid
         Complex/Queue: queue
        Object user id: ouid
        Old validation: oldvalid
        New validation: newvalid
```

The fields are defined as follows:

Field	Description
`Function:` `func`	The request function being audited.
`Violation:` `type_violation`	The type of violation; `None(0)`, `Insufficient privilege(1)`, `Not secadm(3)`, and `Not job owner(4)`.

`Job sequence number: seqno`

The NQS job sequence number or `N/A`.

`Origin host: orighost`

The job submission host or `N/A`.

`Client host: chost`

The client host (that is, the host generating the audit record).

`Origin user id: origuid`

The user ID of the caller.

`Target user id: tuid`

The user ID of the caller.

`Complex/Queue: queue`

The applicable queue name or `N/A`.

`Object user id: ouid`

The user ID of the object (that is, manager being added or deleted through `qmgr`).

`Old validation: oldvalid`

The prior NQS user validation type (`None`, `File`, `Password`, `Password if received`, or `Else file`).

`New validation: newvalid`

The new NQS user validation type (`None`, `File`, `Password`, `Password if received`, or `Else file`).

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t nqscf
May  9 19:31:49 1993  NQS Config. o_lvl: 0 s_lvl: 0 jid:94 pid:6100
r_ids:[operator(9),operator(9)]   e_ids:[root(0),root(0)]  *******
    Login uid: operator(9)

               Function: Abort request(0)
              Violation: Insufficient privilege(1)
    Job sequence number: 13
            Origin host: drizzle
            Client host: drizzle
         Origin user id: operator(9)
         Target user id: operator(9)
          Complex/Queue: N/A
         Object user id: operator(9)
         Old validation: Password
         New validation: Password
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.23 NQS activity record (`SLG_NQS`)

This record is produced whenever a NQS delete request is made. Delete requests include deleting a queued NQS job and sending a signal to an executing job. The signal can be any valid UNICOS signal, including a kill to delete the job signal. These entries are logged through the `slgentry` system call.

If the caller is an NQS administrator, a Cray ML-Safe process activity record is also generated, assuming `SLG_T_PROC` and NQS is configured to enforce MAC rules (that is, `IC_MAC_COMMAND` is enabled).

The `SLG_ACT_NQS` parameter must be on to generate this record. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log NQS activity?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e nqs` command to enable the generation of this record or the `spaudit -d nqs` command to disable the generation of this record.

In addition to the header information, the `SLG_NQS` record displays the following information:

```
Login uid: user_id

           Function: func
          Violation: type_violation
Job sequence number: seqno
        Origin host: orighost
        Client host: chost
     Origin user id: origuid
     Target user id: tuid
Subject compartments: scmp
 Object compartments: ocmp
```

The fields are defined as follows:

Field	Description
`Login uid:` `user_id`	The user ID.
`Function:` `func`	The request function being audited.
`Violation:` `type_violation`	The type of violation; `None(0)`, `Insufficient privilege(1)`, `Not secadm(3)`, and `Not job owner(4)`.
`Job sequence number:` `seqno`	The NQS job sequence number or `N/A`.
`Origin host:` `orighost`	The job submission host or `N/A`.
`Client host:` `chost`	The client host (that is, the host generating the audit record).
`Origin user id:` `origuid`	The user ID of the job's original submitter.

`Target user id:  tuid`

The target user ID.

`Subject compartments:  scmp`

The caller's active compartment set.

`Object compartments:  ocmp`

The executing job's compartment set.

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t nqs
May 10 08:01:12 1993  NQS Activity o_lvl:0 s_lvl:0  jid:32  pid:1246
r_ids:[operator(9),root(0)]   e_ids:[root(0),root(0)] ******
    Login uid: operator(9)

               Function: Remote job delete(155)
              Violation: None(0)
    Job sequence number: 32
            Origin host: squall
            Client host: drizzle
         Origin user id: operator(9)
         Target user id: ce(10)
   Subject compartments: none
    Object compartments: none
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.24 Cray ML-Safe process activity record (SLG_TRUST)

This record is produced by UNICOS Cray ML-Safe processes to describe the Cray ML-Safe activities that are performed on behalf of the user. This record provides a higher-level view of Cray ML-Safe process activity than is provided by normal kernel-level auditing.

Execution of the following Cray ML-Safe processes generates this record: `cleantmp`(8), `cll`(8), `fsoffload`(8), `fuser`(8), `init`(8), `jstat`(1), `labelit`(8), `mail`(1), `mailx`(1), `msgdaemon`(8), `reduce`(8), `reply` (8), `passwd`(1), `ps`(1), `sdss`(1), `setfs`(8), `spnet`(8), `su`(1), and `tpconfig`(8). It is written in NQS when a NQS administrator is allowed to bypass the mandatory access control checking on job deletion and status requests.

Each SLG_TRUST record contains the process name, a list of the permitted privileges for the process, and a brief text description of the the Cray ML-Safe activity performed by the process.

The SLG_T_PROC parameter must be on to generate this record. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log trusted process activity? selection in the UNICOS Installation and Configuration Menu System. Also, for the jstat, sdss, fuser, and ps commands to generate a Cray ML-Safe process activity record, the MAC_COMMAND parameter must be enabled. See Section 8.4.2.6, page 193, for more information.

You can also use the spaudit -e trust command to enable the generation of this record or the spaudit -d trust command to disable the generation of this record.

In addition to the header information, the SLG_TRUST record displays the two following formats of information. The first format is used when no object label information is needed; the second is used when object label information is needed. The second format is generated only by the labelit(8) and setfs(8) commands:

```
Process: process name
permitted privileges: list of process permitted privileges
      privilege text: process privilege text used
      process action: description of process action

Process: process name
permitted privileges: list of process permitted privileges
      privilege text: process privilege text used
minimum compartments: object minimum compartments
maximum compartments: object maximum compartments
       minimum level: object minimum level
       maximum level: object maximum level
              device: device number
      process action: description of process action
```

To display this record, use the reduce command as shown in the following example:

```
$ /etc/reduce -t trust
Jul 15 10:37:11 1993 Trusted Process o_lvl:0 s_lvl:0 jid:1254 pid:34142
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
    Login uid: operator(9)

        Process: spnet
        permitted privileges: PRIV_ADMIN,PRIV_DAC_OVERRIDE,
                              PRIV_MAC_READ,PRIV_MAC_WRITE,
                              PRIV_AUDIT_WRITE,PRIV_AUDIT_CONTROL
              privilege text: TEXT_NULL
              process action: error 17 adding default to NAL

Jul 15 10:37:16 1993  Trusted Process o_lvl: 0 s_lvl:54 jid:1 pid:34156
    r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]       **********
    Login uid: root(0)

        Process: cleantmp
        permitted privileges: PRIV_CHOWN,PRIV_FOWNER,
                              PRIV_DAC_OVERRIDE,PRIV_MAC_UPGRADE,
                              PRIV_MAC_DOWNGRADE,PRIV_MAC_READ,
                              PRIV_MAC_WRITE,PRIV_AUDIT_WRITE,
                              PRIV_AUDIT_CONTROL
              privilege text: TEXT_NULL
              process action: /etc/cleantmp operator /tmp.mld
                              /jtmp/jtmp.001242a
```

```
Jul 15 10:28:50 1993 Trusted Process o_lvl:0 s_lvl:0 jid:1238 pid:33562
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
    Login uid: operator (9)

        Process: ps
        permitted privileges: PRIV_DAC_OVERRIDE,PRIV_MAC_DOWNGRADE,
                              PRIV_MAC_READ ,PRIV_MAC_WRITE,
                              PRIV_AUDIT_WRITE,PRIV_AUDIT_CONTROL
              privilege text: TEXT_NULL
              process action: MAC policy enforced

Jul 15 10:37:16 1993  Trusted Process o_lvl: 0 s_lvl:54 jid:1 pid:34156
    r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]       **********
    Login uid: root(0)

        Process: labelit
        permitted privileges: PRIV_DAC_OVERRIDE,PRIV_MAC_READ,
                              PRIV_MAC_WRITE,PRIV_AUDIT_WRITE,
                              PRIV_AUDIT_CONTROL
             privilege text: TEXT_NULL
        minimum compartments: none
        maximum compartments: comp24 comp39 comp63
               minimum level: 0
               maximum level: 16
                      device: 0
              process action: File system labeled: /dev/dsk/usa
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.25 Use of privilege record (SLG_PRIV)

This record is written to the security log when any system call, except for the `read`(2), `reada`(2), `write`(2), or `writea`(2) system call, invokes a privilege to perform its function or when a system call fails because the calling process lacks a required privilege.

During system call processing, when privileges are checked by the UNICOS kernel, any required active privilege is recorded for that process. Also, for system calls that can only be executed with privilege (for example, the `setsysv`(2) system call), a privilege failure is recorded if the process that issued the system call does not have the necessary privileges.

The SLG_PRIV record is written for the following system calls: access(2), acct(2), acctid(2), adjtime(2), chacid(2), chdir(2), chkpnt(2), chmem(2), chmod(2), chown(2), chroot(2), cpselect(2), creat(2), dacct(2), exec(2), fchmod(2), fchown(2), fcntl, fgetpal(2), fjoin(2), fpathconf(2), fsecstat(2), fsetpal(2), fstat, getdents(2), getfacl, getpal(2), getsysv(2), getusrv(2), ialloc(2), ioctl(2), jacct(2), join(2), lchown(2), limit(2), limits(2), link(2), listio(2), lsecstat(2), lstat(2), mount(2), nice(2), nicem(2), open(2), pathconf(2), plock(2), ptyrecon(2), quotactl, readlink(2), rename(2), restart(2), resume(2), rmdir(2), rmfacl(2), schedv(2), secstat(2), select(2), setdevs(2), setfacl(2), setfflg(2), setgid(2), setgroups(2), setjob(2), setlim(2), setpal(2), setpermit(2), setregid(2), setreuid(2), setsysv(2), settimeofday(2), setucmp(2), setuid(2), setulvl(2), setusrv(2), stat(2), stime(2), suspend(2), tabinfo(2), tabread(2), target(2), trunc(2), ulimit(2), umount(2), unlink(2), upanic(2), utime(2), and wracct(2).

Many system calls can execute without using privilege if the user is not attempting to override a system security policy restriction. These same system calls may also be made by a privileged process that needs to override such restrictions. Because of this situation, privilege failure records are not issued for most system calls. Instead, the failed attempt to perform the requested function is logged as another type of record such as mandatory access failure or discretionary access failure.

The SLG_PRIV record is written when system call processing completes, but before control is returned to the calling process. At this time, a check is made to determine if any privileges were used by the system call or if any privilege failures were recorded. If privileges were used or privilege failures occurred, then a SLG_PRIV record is written.

The SLG_PRIV parameter must be on to generate this record. To set this parameter, use the Configure system -> Multilevel security (MLS) configuration->Security log file configuration->G selection in the UNICOS Installation and Configuration Menu System.

You can also use the spaudit -e priv command to enable the generation of this record or the spaudit -d priv command to disable the generation of this record.

The format is the same as described for the discretionary access violation record, with the addition of the following fields. See Section 8.8.7.6, page 280, for a description of the format. The fields are defined as follows:

Field	Description
`privileged used`	List of privileges that were actually used to perform the described function.
`privilege failed`	List of privileges that are needed by subject, but the subject did not have to perform the given function.

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t priv
Jul 15 10:41:49 1993  Privilege Use o_lvl: 0 s_lvl: 0 jid:1259 pid:34239
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),mail(1)]
**********
    Login uid: operator(9)

   Function: stat (147)     Violation: NONE (0)
  Subject: Active Compartments : none
            Valid Compartments : none
                   Permissions : suidgid
                         Class : 0
             Active Categories : none
              Valid Categories : secadm sysadm
                privilege used : PRIV_MAC_READ
              privilege failed : none

Jul 15 10:41:49 1993  Privilege Use o_lvl: 0 s_lvl: 0 jid:1259 pid:34239
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),mail(1)]
**********
    Login uid: operator(9)

   Function: secstat (92)     Violation: NONE (0)
  Subject: Active Compartments : none
            Valid Compartments : none
                   Permissions : suidgid
                         Class : 0
             Active Categories : none
              Valid Categories : secadm sysadm
                privilege used : PRIV_MAC_READ
              privilege failed : none
```

See Section 8.8.8, page 343, for more information on using this command.

8.8.7.26 Cray/REELlibrarian (CRL) activity record (`SLG_CRL`)

There are four types of Cray/REELlibrarian (CRL) activity log records. Each type is defined by the process that logs the record and the class of activity being monitored. The CRL processes that generate the four types are as follows: `tpdaemon`(8), `spset`(1), the CRL daemon, and the CRL client commands. Each CRL record contains a field that defines which of the four processes requested the log entry.

The `tpdaemon` logs access mediation of the tape volumes and the tape files. Access requests are for the information that is resident on the volumes (that is, the tape file contents).

The `spset`(1) command logs the use or attempted use of security administrator privilege to change the security label of a CRL file or volume set.

The CRL daemon logs the following three different classes of catalog access:

- The creation and deletion of volume sets and files
- The requests that change the mandatory access control and discretionary access control attributes of volume sets and files
- The use of CRL administrator privilege.

The CRL client commands (defined as those commands linked to `/bin/RCOM`) log user access of CRL objects (files and volume sets). When appropriate, the client process is specific in its identification of the object whose access is being requested. In other cases, where the CRL request is not object-specific, the user request description is logged.

The `SLG_LOG_CRL` parameter must be on to generate this record. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log Cray/REELlibrarian activity?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e crl` command to enable the generation of this record or the `spaudit -d crl` command to disable the generation of this record.

In addition to the header information, the `SLG_CRL` record displays the following information:

```
          Subject: Label : level,cmpt
      caller privileges : priv
    requested operation : req_oper
                 result : result
           entry logger : logger
          Object: Label : level,cmpts
```

The fields are defined as follows:

Field	Description
`Subject: Label : level,cmpt`	This is the active level and compartments of the subject requesting the CRL operation.
`caller privileges : priv`	The active privilege set of the caller at the time of the request. This field is valid only for the CRL daemon entries, as the daemon is not aware of the caller's privileges. The CRL daemon assumes the Cray ML-Safe client making the request has validated the requester's privileges and that the client logs the application of any such privileges.
`requested operation : req_oper`	An ASCII description of the CRL operation that was requested. It may take the form of a CRL client/server interface subroutine name (for example, `rl_vid`), a specific CRL daemon object operation (for example, delete), or a generic administrator privilege use message which is quantified by the request parameters field.
`result : result`	An ASCII description of the error that was encountered or `request complete` if no error occurred.
`entry logger : logger`	Describes which process is making the CRL activity log entry.

`Object:  Label :  level, cmpts`

The level and compartment set of the file or volume set in a specific CRL object reference. In the case of a volume set, it is the lower label of the (possibly) multilevel object.

For a specific volume set object reference, the following fields are used:

Field	Description
`volset name`	The volume set name in valid CRL syntax.
`volset upper level`	The upper level of the volume set. The lower level of the (possibly) multilevel volume set is found in the `Object: Label:` field.
`volset upper compartments`	The upper compartment set of the volume set. The lower compartment set of the (possibly) multilevel is found in the `Object:  Label:` field.
`request parameters`	An ASCII string which qualifies the `requested operation` field to help determine the effect of the request.

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t crl
Oct 12 16:22:19 1993  REELlibrarian o_lvl:0 s_lvl:0  jid:135  pid:12066
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
   Login uid: operator(9)

                    Subject: Label : 0, none
                 caller privileges : none
               requested operation : delete
                            result : request complete
                      entry logger : crl daemon
                       volset name : .SL0007
                volset upper level : level0
         volset upper compartments : none
                     Object: Label : 0, none



Oct 12 16:22:19 1993  REELlibrarian o_lvl:0 s_lvl:0  jid:126  pid:21990
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
   Login uid: operator(9)


                    Subject: Label : 0, none
                 caller privileges : none
               requested operation : rl_vscratch
                            result : request complete
                      entry logger : crl client
                request parameters : .SL0007
```

```
Oct 15 16:05:39 1993  REELlibrarian o_lvl:0 s_lvl:0  jid:64  pid:7618
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
   Login uid: operator(9)


                    Subject: Label : 0, none
                 caller privileges : none
               requested operation : CRL administrator privilege
                            result : request complete
                      entry logger : crl daemon
                request parameters : rl_nvrec request


Oct 15 08:53:10 1993  REELlibrarian o_lvl:0  s_lvl:0  jid:64  pid:6902
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
   Login uid: operator(9)


                    Subject: Label : 0, none
                 caller privileges : none
               requested operation : delete
                            result : request complete
                      entry logger : crl daemon
                         file name : .SL0A09^FILE2
                     Object: Label : 0, none
```

```
Oct 15 10:18:19 1993  REELlibrarian o_lvl:0  s_lvl:0  jid:64  pid:6902
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
   Login uid: operator(9)


                    Subject: Label : 0, none
                 caller privileges : none
               requested operation : CRL administrator privilege
                            result : request complete
                      entry logger : crl daemon
                request parameters : 'x' mode permission



Oct 15 10:18:41 1993  REELlibrarian o_lvl:3  s_lvl:0  jid:60  pid:91261
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
   Login uid: operator(9)


                    Subject: Label : 0, none
                 caller privileges : none
               requested operation : rl_vid
                            result : mandatory violation
                      entry logger : crl client
                       volset name : .SL0021
                volset upper level : level3
         volset upper compartments : comp24
                     Object: Label : 3, comp24
```

8.8.8 The `reduce` command

The `reduce`(8) command extracts and formats entries collected by the security log daemon and generates a report of the data. The default is to report all entries currently in the disk-resident security log (you can select one of the date-defined files to be processed instead of the current log).

On a system with `PRIV_SU` enabled, you must be `root` (UID = 0) to use this command.

On a system using the PAL-based privilege mechanism, you must have an active `secadm` category to use the `reduce` command.

By using one or more of the `reduce` options, you can choose the type of event to be extracted and formatted, as explained in the following sections. See the `reduce` man page in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022, for more information on all the options.

8.8.8.1 Selecting record types (`-t` option)

The `-t` option allows you to specify the type of entry formatted. The valid security record types are as follows:

Type	Description
`go`	System startup
`stop`	System shutdown
`tchg`	System time change
`cchg`	System configuration change
`dac`	Discretionary access change
`disc`	Discretionary access violation event
`mand`	Mandatory access event
`oper`	Not currently used
`logn`	Login validation process
`netw`	Not currently used
`disk`	Not currently used
`ssd`	Not currently used
`eoj`	End of job
`chdir`	Change directory
`tape`	Tape I/O error
`secsys`	Security-related system calls
`nami`	File manipulation system calls
`setuid`	`setuid` system calls
`other`	Not currently used
`sulog`	`su` attempts
`netip`	Network IP layer
`nfs`	Cray-to-Cray NFS requests

`xfer`	File transfers
`netcf`	Network configuration changes
`audit`	Security auditing criteria changes
`nqscf`	NQS configuration changes
`nqs`	NQS activity
`trust`	Cray ML-Safe process activity
`priv`	Use of privilege
`crl`	Cray/REELlibrarian (CRL) activity

See the previous sections on auditing records for examples of using the `-t` option.

8.8.8.2 Printing security labels in record header (`-S` and `-L` options)

On UNICOS systems, you can print both the security label of the subject and object in the record header by using the `reduce -S` option.

The subject's security label information appears in the following format in the header:

`S_Label:` *level*`,`*comp1*`,`*comp2*`,...,`*compn*

The object's security label information appears in the following format in the header:

`O_Label:` *level*`,`*comp1*`,`*comp2*`,...,`*compn*

When then the `-S` option is used, no subject or object label information is printed in the record body. The format produced by the `-S` option will become the default format in a future UNICOS system release.

The `reduce -L` option prints only the subject's and object's security level as part of the record header. This option has been maintained for compatibility with previous UNICOS releases, but is no longer supported.

8.8.8.3 Selecting records by object label (-O option)

You can use the `reduce -O` option to select record entries by the security label of an object. The security label is specified in the following form:

level`[,`*compartment*`[,`*compartment*`[...]]]`

The following example shows what is displayed when the `-O` option is used; only those records that have the same object label (that is, with a label of security level 4 and compartment `comp24`) are selected. Also, the new version of the record header is displayed, as the `-S` option is used:

```
$ reduce -s11231255 -e11231300 -p -S -O 4,comp24
Nov 23 12:55:42 1993  Mandatory      jid:59  pid:5433
r_ids:[operator(9),operator(9)]  e_ids:[operator(9),operator(9)]
    S_Label: 0,none  O_Label: 4,comp24
    Login uid: operator(9)

   Function: stat (147)     Violation: Permission denied (13)
             System call : stat (147)
   Subject: Permissions : suidgid
                   Class : 0
              Categories : none
             Access Mode : read
   Object: uid: operator(9)  gid: operator(9)  device: 34, 5
                                    inode: 34
                Pathname : /dev/ttyp000  * No current directory *
                   Class : 0
              Categories : none
                    Mode : 20622



Nov 23 12:55:47 1993  Mandatory  jid:13  pid:5458
r_ids:[operator(9),operator(9)]  e_ids:[operator(9),operator(9)]
    S_Label: 0,none  O_Label: 4,comp24
    Login uid: operator(9)

   Function: setfcmp (86)     Violation: NONE (0)
             System call : setfcmp (86)
   Subject: Permissions : suidgid
                   Class : 0
              Categories : none
             Access Mode : write
   Object: uid: operator(9)  gid: operator(9)  device: 34, 22  inode: 693
                Pathname : /tmp.mld/jtmp/jtmp.000013a.mld/000/mxc.5308
                            /mxctcs02_dir/reg_file  * No current directory *
                   Class : 0
              Categories : none
                    Mode : 100770
```

8.8.8.4 Displaying path names (-p option)

The `-p` option reconstructs a path name. The path tracking logging option must be enabled in order for this option to work; see Section 8.8.6, page 270, for more information on this option. For each entry that contains a relative path name, this option attempts to reconstruct the full path name being referenced from the previous path history.

For displaying record types that record object-relevant functions, such as the discretionary and mandatory access records, it is helpful to use the `-p` option. If this option is not used, the only file information displayed is the inode number, and major and minor device numbers, making locating the actual file name time consuming.

8.8.8.5 Tracking a specific user name (`-l` and `-u` options)

The `-l` and `-u` options of the `reduce` command allow you to trace a specific user. The `-l` option tracks a login user ID from the point of login throughout the session. The login user ID is the most reliable identifier in the log entry because it cannot be changed after logging in. This ID is the one for which the password was checked.

The `-u` option can be used to monitor a login account when a user logs into the account or a `setuid` call is made to the account.

For example, a user may log in using the name `john`. Then the user could use the `su` command to change the user name to `ted` and then to `beth`. If you want to see all the activity for user login sessions under the login `john`, then the `-l` option should be used.

If you wanted to see all activity done under the user login `john` (which would include any `setuid` calls to the login `john`), then the `-u` option should be used.

If a user creates a new session with a different real user ID, using the `-l` option does not work for tracking during the new session. An administrator can tell when a new session is created by a `setsid (117)` in the `Function` field of the security system call audit record (`SLG_SECSYS`). To track the new session, use the `-l` option to track the new user ID and/or the `-j` option to track the new job ID. The `-j` option is described in the next section.

When tracking sublogin sessions, the records contain either the initial login user ID or the second login user ID. The job ID should be used to track records for sublogins.

The following example shows activity traced for the login `rll`. In this example, `rll` never logged in during the time frame. Rather, the login `jnk` executed under the real user ID of `rll`:

```
$ /etc/reduce -u rll

Apr 29 11:04:12 1991  Setuid Syscall o_lvl:0  s_lvl:0  jid:85 pid:9464
    r_ids:[rll(626),tng(23510)]   e_ids:[rll(626),tng(23510)] ******
****
    Login uid: jnk(204)

   Setuid call from root (0) to rll (626) was successful



Apr 29 11:04:12 1991  Discretionary o_lvl:63  s_lvl:0  jid:85 pid:9464
    r_ids:[rll(626),tng(23510)]   e_ids:[rll(626),tng(23510)] ******
****
    Login uid: jnk(204)

   Function: open (5)     Violation: Permission denied (13)
   Subject: Compartments : none
             Permissions : suidgid
                   Class : 0
              Categories : none
             Access Mode : write
   Object: Level: 63  uid: jnk(204) gid:tng(23510) device:0, 211
   inode: 26
            Compartments : none
                   Class : 0
              Categories : none
                    Mode : 100644
```

8.8.8.6 Tracing a user's login session (-j option)

Every user session is assigned a unique job ID at the time of login. This ID can be used to pick out all related activity of a single interactive or batch session by using the `-j` option of the `reduce` command. The following example shows how to use the `-j` option:

```
$ /etc/reduce -j 85

Apr 29 09:05:30 1991 Security Syscall o_lvl:0  s_lvl:0 jid:85 pid:2327
  r_ids:[jnk(204),tng(23510)]   e_ids:[jnk(204),tng(23510)] **********
Login uid: jnk(204)

  Function: setusrv (89)     Violation: NONE (0)
  Subject: Active Compartments : none
            Valid Compartments : none
                   Permissions : suidgid
                         Class : 0
             Active Categories : none
              Valid Categories : secadm



Apr 29 09:05:37 1991 Setuid Syscall  o_lvl:0  s_lvl:0  jid:85 pid:2327
  r_ids:[jnk(204),tng(23510)]   e_ids:[jnk(204),tng(23510)]
*********
Login uid: jnk(204)

   Setuid call from root (0) to jnk (204) was successful

Apr 29 09:05:37 1991 Change Directory o_lvl:0  s_lvl:0  jid:85 pid:2327
  r_ids:[jnk(204),tng(23510)]   e_ids:[jnk(204),tng(23510)] **********
Login uid: jnk(204)

    Relative directory path: /sn131/mktg/tng/jnk
    Absolute directory path: /sn131/mktg/tng/jnk


Apr 29 09:05:48 1991 Setuid Syscall  o_lvl:0  s_lvl:0  jid:85 pid:2344
  r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)] **********
Login uid: jnk(204)

   Setuid call from jnk (204) to root (0) was successful
```

```
Apr 29 09:10:08 1991  Mandatory    o_lvl:0  s_lvl:0  jid:85 pid:2861
  r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)] **********
Login uid: jnk(204)

   Function: open (5)     Violation: Security category violation (319)
   Subject: Compartments : none
             Permissions : suidgid
                   Class : 0
              Categories : none
             Access Mode : read
   Object: Level: 0  uid: root(0) gid:sys(3) device:0, 245 inode:262
            Compartments : none
                   Class : 0
              Categories : none
                    Mode : 104755


Apr 29 11:04:12 1991  Setuid Syscall o_lvl:0  s_lvl:0  jid:85 pid:9464
  r_ids:[rll(626),tng(23510)]   e_ids:[rll(626),tng(23510)] *********
Login uid: jnk(204)

   Setuid call from root (0) to rll (626) was successful


Apr 29 11:04:12 1991  Discretionary  o_lvl:63  s_lvl:0  jid:85 pid:9464
  r_ids:[rll(626),tng(23510)]   e_ids:[rll(626),tng(23510)] **********
Login uid: jnk(204)

   Function: open (5)     Violation: Permission denied (13)
   Subject: Compartments : none
             Permissions : suidgid
                   Class : 0
              Categories : none
             Access Mode : write
   Object: Level: 63  uid: jnk(204)  gid: tng(23510)  device: 0, 211
   inode: 26
            Compartments : none
                   Class : 0
              Categories : none
                    Mode : 40700
```

```
Apr 29 11:04:31 1991  Mandatory    o_lvl:0  s_lvl:0  jid:85 pid:9493
  r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)] **********
Login uid: jnk(204)

   Function: open (5)     Violation: Security category violation (319)
   Subject: Compartments : none
             Permissions : suidgid
                   Class : 0
              Categories : none
             Access Mode : read
   Object: Level: 0  uid: bin(2)  gid: bin(2)  device: 0, 245
   inode: 267
            Compartments : none
                   Class : 0
              Categories : secadm
                    Mode : 100755
```

8.8.8.7 Reducing security log input (`-r`, `-R`, and `-f` options)

The `-r` option of the `reduce` command can be used to save the selected security log entries in raw format. The `-R` option performs the identical function, but while the `-r` option uses a default file to save the entries, you must specify an output file for the `-R` option. The following example shows how to use the `-R` option:

```
$ /etc/reduce -t logn,mand,disc  -R slog.4_1 > /dev/null
$ ls -l slog.4_1
-rw-------   1 root     root       16544 Apr  2 00:01 slog.4_1
```

If a redirect to `/dev/null` is not done (as shown in the previous example), `reduce` echos all of the records that are being placed into the raw log.

The resultant file can be viewed later by using the `reduce -f` command. This file could then be moved offline to another storage medium to conserve space.

8.8.9 Monitoring security-relevant events

The `spcheck`(8) command checks a variety of security-relevant events on a UNICOS system. It should be used on a daily basis to detect actual or potential security breaches.

The `-a`, `-g`, `-l`, `-p`, `-q`, and `-w` options are of special importance when checking a UNICOS system:

- The `-a` option reports users who have the administrative category and/or permissions (you need both `root` permission and the `secadm` category to use this option).
- The `-g` option reports users in groups `root`, `adm`, `bin`, and `sys`; the `-G` option does the same thing, but uses the group names listed in `/etc/grpcheck`. The `/etc/grpcheck` file contains valid group names and has the following format:

  group1 group2 group3 ... groupn

  You need `root` permission to use this option.
- The `-l` option lists any user login IDs that have not logged in for at least 14 days or 180 days. Also, it lists `amp;.profile` and `amp;.cshrc` files that can be written by anyone (you need `root` permission and the `secadm` category to use this option). See the following example.
- The `-p` option reports users with duplicate IDs, users who cannot change their password, and users whose passwords do not expire (you need `root` permission and the `secadm` category to use this option).
- The `-q` option reports users who have 10 or more `su` command failures in the current `sulog` file (you need `root` permission to use this option).
- The `-w` option reports files in the system that can be written by any system user.

This command has some options that can be used by users other than the security administrator. See the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022, for more information on the command and its options.

The following example shows how to use the `spcheck` command:

```
$ /etc/spcheck -l

 =====Users with no login for 14 days =====
         root
         sync
         bin
         sys
         adm
         cron
         nqs
         uucp
         ce
         ttg
         lms
         mac

=====Users with no login for 180 days =====
         llm
         pim
         mcm

=====Users with .profiles or .cshrc writable by anyone =====
         cet
         kan

$ /etc/spcheck -q

 =====User’s with excessive su failures in /usr/adm/sulog=====

        jnn   had 10 su failures on 2/22
```

8.8.10 Security violation error codes

Table 11 describes the return codes that are set in the security log to indicate a security policy violation.

Table 11. Security violation error codes

Error condition	Error code	Description
ESYSLV	300	Indicates the specified security level falls outside the allowed security level range of the process, file system, or UNICOS system.
EREADV	301	Indicates an attempt to gain read access to a file failed because the user's active security level was less than the file's security level.
EWRITV	302	Indicates an attempt to gain write access to a file failed because the user's active security label was not equal to the file's security label.
EEXECV	303	Indicates an attempt to gain execute/search access to a file failed because the user's active security level is less than the file's security level.
ECOMPV	304	Indicates the specified security compartment falls outside the allowed security compartment range of the file.
EMANDV	305	Indicates a mandatory access violation.
EOWNV	306	Indicates that the user is not the owner of the object to be accessed.
ELEVELV	307	Indicates that the requested security level is outside the user's security-level range.
ESECADM	308	Indicates that a subject who does not possess the proper authorization attempted to perform an operation available only to properly authorized users.
EFLNEQ	309	Indicates that a security level violation occurred during the mounting of a file system.
ENOTEQ	310	Indicates buffer level is not equal to the file level.
EPERMIT	311	Indicates a permission violation; the appropriate permission is required.
EACLV	312	Indicates an ACL access violation.
ENOACL	313	Indicates that no ACL list was found.

Error condition	Error code	Description
`ESLBUSY`	314	Indicates that an attempt to open the security log (`/dev/slog`) failed because pseudo device `/dev/slog` was already open to another process.
`ESLNXIO`	315	Indicates an attempt to open the security log (`/dev/slog`) for other than a read operation.
`ESLFAULT`	316	Indicates a read error on the security log (`/dev/slog`).
`ESLNOLOG`	317	Indicates that the security log state was `OFF` when a security log request was made.
`EINTCLSV`	318	Indicates the requested integrity class falls outside the allowed integrity class range of the process, or the UNICOS system. The class value is no longer used.
`EINTCATV`	319	Indicates the requested category is not included in the allowed categories of the process, or the UNICOS system.
`ENONAL`	320	Indicates that no network access list (NAL) entry was found.
`EMNTCMP`	321	Indicates that a security compartment violation occurred when a mount file system request was processed.
`EFIFOV`	322	Indicates a security FIFO violation.
`EAPPNDV`	323	Indicates that a security violation occurred on an append request.
`ETFMCATV`	324	Indicates user requested an illegal combination of categories
`ECOVERT`	325	Indicates a user performed a legal operation that has created a possible covert channel condition.
`ERCLSFY`	326	Indicates an attempt has been made to change the security label of a file that does not reside in a wildcard directory.
`EPRLABEL`	327	Indicates that security label printing is disabled.
`ENONSECURE`	328	Indicates that a MLS system call was made on a system running a non-MLS kernel.
`ESECFLGV`	329	Indicates a request to set file security flags has failed because the requested flags are not allowed for the system.
`EHOSTNAL`	330	Indicates access to or from an unauthorized host or workstation was attempted. The host is not authorized in the NAL.

Error condition	Error code	Description
ESLVLNAL	331	Indicates a security level was detected outside of the range of security levels authorized for the host in the NAL. The kernel detected this condition when processing the Internet Protocol (IP) security option associated with a datagram.
ESCMPNAL	332	Indicates a security compartment was detected outside of the range of compartments authorized for the host in the NAL. The kernel detected this condition when processing the IP security option associated with a datagram.
EMODENAL	333	Indicates an illegal mode (send or receive) of transfer was attempted by a host or workstation.
ESLVNIF	334	Indicates a security level was detected outside of the range of security levels authorized for the UNICOS network interface (I/F). The kernel detected this condition when processing the IP security option associated with a datagram.
ESCMPNIF	335	Indicates a security compartment was detected outside of the range of security compartments authorized for the host in the UNICOS I/F. The kernel detected this condition when processing the IP security option associated with a datagram.
ESOCKLVL	336	Indicates an illegal attempt was made to change the security level of a single level socket (SLS) connection. Except for a privilege granted by the security administrator (for example, NFS), all socket connections are created as SLS.
ESOCKCMP	337	Indicates an illegal attempt was made to change the security compartment of a SLS connection. Except for a privilege granted by the security administrator (for example, NFS), all socket connections are created as SLS.
ENFSAUTH	338	Indicates the proper authentication credentials were not passed to NFS.
ESLVLNRT	339	Indicates a security level was detected outside of the range of security levels authorized for the network route selected, or a route with the correct sensitivity label could not be found.
ESCMPNRT	340	Indicates a security compartment was detected outside of the range of security compartments authorized for the network route selected, or a route with the correct sensitivity label could not be found. The kernel detected this condition when selecting routes.

Error condition	Error code	Description
EBADIPSO	341	Indicates an illegal IP security option was detected by the kernel. The kernel performs integrity and security checks against each IP security option received.
ENOIPSO	342	Indicates an IP security option did not accompany an incoming datagram. The kernel detects this condition by using IP security option information defined in the NAL for each host/workstation.
ESLVLMAP	343	Indicates a translation error was detected when mapping the security level (between UNICOS form and network form). When necessary, using the NAL, the kernel translates the UNICOS security label to the network security label (for outgoing datagrams) and translates the network security label to the UNICOS security label (for incoming datagrams).
ESCMPMAP	344	Indicates a translation error was detected when mapping the security compartment (between UNICOS form and network form). When necessary, using the NAL, the kernel translates the UNICOS security label to the network security label (for outgoing datagrams) and translates the network security label to the UNICOS security label (for incoming datagrams).
EAUTHFLG	345	Indicates an authority protection violation was detected for either an incoming or outgoing datagram with a Basic Security Option.
EIPSOMAP	346	Indicates no translation table was available to translate the security label for a given host connection. The kernel could access the mapping table identified in the NAL for the host.

8.9 NQS operations

For more information on using NQS on a UNICOS or Cray ML-Safe system configuration, see the *NQE Administration*, Cray Research publication SG–2150.

8.10 Tape operations

For more information on using tapes on a UNICOS or Cray ML-Safe system configuration, see *Tape Subsystem Administration*, Cray Research publication SG–2307.

Note: If your site plans to allow nonadministrative user access to tapes on a Cray ML-Safe system tapes, Cray/REELlibrarian (CRL) is required. CRL is not required on a Cray ML-Safe system if administrative-only access is allowed.

Sites are not required to run tapes with a Cray ML-Safe system configuration; in this case, CRL is not required. CRL is licensed and charged separately from the UNICOS system. See the *Cray/REELlibrarian (CRL) Administrator's Guide*, Cray Research publication SG–2127, and the *Cray/REELlibrarian (CRL) User's Guide*, Cray Research publication SG–2126, for more information on using CRL on UNICOS and Cray ML-Safe system configurations.

8.11 TCP/IP operations

For information on TCP/IP operations on a UNICOS system with the MLS feature, see the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304.

8.12 UNICOS NFS operations

For more information on NFS operations on a UNICOS system with the MLS feature, see the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304.

8.13 MLS data migration operations

The data migration facility supports the UNICOS and Cray ML-Safe system configurations. See the *Cray Data Migration Facility (DMF) Administrator's Guide*, Cray Research publication SG–2135, for more information.