# Cray Security Administration: Tricks of the Trade

**Bonnie L. Hall**

**Exxon Upstream Technical Computing**

**1997 Cray User Group Meeting, San Jose CA.**

## Abstract

*Control is an important aspect of computer security. There are many things an administrator can do to improve system security without installing expensive tools or waiting for vendor enhancements. This paper will focus on the control measures which may be implemented on a large UNIX system to help the administrator gain control over users, data and privilege. While the issues presented in this paper are ones which are more evident in a large computer system, many of the suggestions and examples herein are relevant to any UNIX machine. References to Cray UNICOS Operating System specific tools such as System Audit Logs, Privilege Access Lists, and NFS UID/GID mapping, will be noted.*

## Security & Controls

The topic of computer security is a huge one, encompassing all aspects of computing: hardware, software, networks, operating systems. This paper will narrow the broad topic of security to just one aspect: controls.

The term "controls" is used to refer to policies, procedures and tools which are used to gain control over a computer system. Every system administrator strives to find the proper balance of control for his installation. Too many controls and the system becomes bogged down and difficult to use. Too few controls and the system becomes difficult to administrator and secure.

It is essential for a system administrator to be in control of the system if it is to be secure. There is a lot more to securing a system then just the implementation of good control measures, but reasonable controls will create a firm foundation for a secure computing system.

Many sites have years of experience administering small UNIX machines which are shared among several people working on a common project. In this environment the emphasis of security has been on the network and keeping unauthorized users off the system. As workstations grow into superservers and multiprocessors, the user community becomes larger and more diverse. Large, expensive systems frequently manage valuable assets. These factors bring control concerns from the old mainframe model into the limelight in the client/server arena. Data is not intended to be shared among the entire user community, it becomes difficult keep track of the users, and the use of privilege comes under more scrutiny.

The intention of this paper is not to reiterate existing documentation nor to recap information which can be found in a typical UNIX security book. Instead it will focus on things an administrator can do to organize a system in a sensible and efficient manner. This paper will discuss controls which are sensible to implement on any system, whether or not the site has chosen to implement Cray UNICOS security features such as data compartments and Privilege Access Lists (PALs).

## Risk Assessment

To implement appropriate controls for a given environment it is necessary to assess risk; determine what needs to be protected and how much should be spent in the protection. While it is seldom feasible to eliminate all risk from a system, cost effective controls can be implemented to reduce risk to an acceptable level. There are many factors to consider when determining possible exposures on a system, including (but not limited to):

- Health and Safety Concerns
- Legal and Ethical Issues
- Information Disclosure
- Information Loss
- Disruption of Service

Risk is inherent to all business. The only time when there is no risk is when there is nothing worth protecting, which is contrary to the notion of business itself. There are two components to risk: severity and probability. It is possible to reduce the risk of an exposure to an acceptable level by lowering the severity of the exposure, the probability of the exposure, or both.

The first component of risk, severity, may be viewed as the cost incurred to business if a vulnerability is exploited. It is important to realize that cost is not merely a monetary issue. Any types of loss should be viewed as cost, including wasted work force effort, a tarnished reputation or loss of competitive advantage. If the consequences from an exploited exposure are severe, it is necessary to reduce the risk.

The second component of risk is probability, how likely is it that an exposure will be exploited? If an exposure is likely to occur frequently, it is often wise to reduce the risk of the exposure.

The cost of implemented controls must be less than the potential loss from the exposure. It is up to the technical experts to identify potential exposures and the probability of occurrence. It is the responsibility of the system owners, the people who own the assets which are at risk, to determine how much risk is acceptable and how much to spend to reduce it. Input from the system owner is essential in the implementation of reasonable controls. A good example is the risk of a system crash. An administrator's first reaction is often "UNACCEPTABLE!!" because he feels that a system crash is a poor reflection of his technical expertise. However, the system owner is likely to view a system crash in a different light. If the "cost" of a crash is negligible in relation to the value of the business, the system owner will not be willing to spend a lot of time or money in preventing an occasional occurrence of the exposure.

For the purpose of illustration, invent a scenario of a car manufacturer designing an automatic braking system. Imagine that a risk assessment revealed that a sealant used in the brakes may become brittle and fail in temperatures below minus 20°F. The exposure is brake failure. The probability is low, not many people drive in minus 20°F. But the severity is high, the worst case would be the loss of human life. An inexpensive control, say warning people not to drive in extreme cold, would not good enough in this case. Due to the severity of the exposure, the system owner would most likely determine that it is necessary to spend more money on controls and attempt to eliminate the risk completely.

Often common sense control measures are often sufficient to reduce risk to a reasonable level. For example, years ago I rented an apartment in a good neighborhood and did not concern myself with the risk of robbery. I left a large jar of small change in full view of a window, even though I was storing jewelry and coins of value in the apartment. One afternoon I returned from work to find my window smashed and the change jar missing. I was lucky and the change jar was the only item taken, the severity of the robbery could have been much worse. Lack of a inexpensive control measure, moving the change jar away from the window, nearly resulted in a very costly security breach.

In summary, a risk assessment is a tool to help system administrators and system owners determine reasonable control measures to protect the system from identified exposures. The process of risk assessment may be summed up as follows:

1. Determine system exposures. "What can go wrong?"
2. Evaluate the severity of each exposure. "What sort of damage could result?"
3. Determine the probability of each exposure. "How likely is this to occur?"
4. Determine control measures to reduce risk. "What can I do to reduce the likelihood of this occurring?"
   "What can I do to reduce the severity of an occurrence?"
5. Evaluate cost of each control measure. "Is the cost of the control less than the potential cost of exposure?"
6. Plan to implement controls which make sense. "Have I reduced the risk to an acceptable level?"

This paper will examine potential control measures to reduce risk in three general areas: data, users and privilege.

## Managing Data

### Data Ownership

On a large system it becomes difficult for an administrator to keep track of all the data and who should be allowed access to it. A good way to manage this problem is to assigns data owners for each UNIX group on the system. Since all data has a group associated with it, giving the groups owners results in all data having an owner. The administrator can keep a text file with this mapping and use it to manage all data on the system. The data owner is responsible for knowing what data is stored on the system and who can access it. The administrators runs reports to help data owners review and maintain access to their data but are not responsible for knowing the contents of each file and who should be allowed to look at it.

Sample file to map groups to data owners:

```
#group:       owner:       organization:  mailstop:
research:     Jane Doe:    R&D:           123 Bldg. A:
testing:      John Smith:  Systems:       999 Bldg. Q:
production:   Jill Smith:  Systems:       111 Bldg. Q:
operations:   A. Jones:    Operations:    100 Bldg. Z:
```

The awk command may be used to create list of groups in an organization. Let $grpown be set to the group ownership file:

```
getgrps ()
{ cat $grpown |\
  awk -F: -v org=$1 '{
    if ($3==org)
     { grprpt[i]=$1
       i++}}
  END{ for ( i in grprpt ) print grprpt[i]}' }
```

With this routine it is easy to create a report for each organization:

```
for org in `cut -d: -f3 $grp.own | sort -u`
  do for group in `getgrps $org`
    do
    processing for each group
  done
done > $org.rpt
```

In most cases the user of the data is not a good choice for the data owner. The owner must be in a position of responsibility and be empowered to make business decisions with a cost equivalent to the data's value. Therefore management is often the best person to be a data owner.

Each group of customers will have a procedure to follow to grant data access. The data owners may want to appoint representatives to follow procedures and give approval on their behalf. This is fine, as long as the system administrators knows who is the proper contact for each group on the system. The key point is that each user organization takes responsibility for their data and the system administrator is not concerned with the details of how each customer decides to manage their data. The system administrator can follow a single set of procedures for satisfying all user requests:

- Receive request from the user community
- Obtain authorization from data owner or appointed representative
- Satisfy request using approved and documented mechanism
- Log actions for subsequent report to data owners

**Data Access Review**

System Administrators may wish to scan for permission changes on key files on a regular basis. This can be done easily with the UNICOS spfilck command. The spfilck command monitors and controls ownership, mode, and security parameters of files. It reads or creates a control file, /etc/permlist, which defines the actual or desired file attribute (owner, mode, security) definition. If file permissions change from the desired values, the spfilck command will change the file permission back to the desired value and list the changes so an administrator may investigate.

In addition to the frequent scans for changes to key file permissions, the data owners should perform a more comprehensive data access review at least annually. A script may be written that uses the group ownership file as input and scans the system to create a report for each data owner. This review should be customized to scan for items which are important to the installation. The following items may be of interest:

For each data owner
- For each group he owns
    1. Permissions on high level directories
    2. Permissions on all files in key directories
    3. Data with world write access
    4. Contents of Access Control Lists
    5. Permission on SUID/SGID executables
    6. NFS mounted file systems
    7. Exported file systems
    8. For each member of the group
        a. person's name and organization
        b. contents of trusted hosts file
        c. permissions on home directory
        d. date of last system access
        e. special privileges assigned to user

**Data Access Violation Reporting**

A data owner may want to see all data access violations committed against their data. The UNICOS Install tool provides parameters to set for logging of all data access violations. It is important to set the parameter to track all path names on access as well to record the full name of the violated file in the log. The UNICOS reduce command is used to obtain log information. A data access violation entry would look something like this:

```
Apr 21 08:03:29 1997 Discretionary o_lvl: 0 s_lvl: 0 jid:57163 pid:42017
r_ids:[user1(12345),group1(100)] e_ids:[user1(12345),group1(100)]*******
Login uid: user1(12345)

Function: open (5)     Violation: Permission denied (13)
System call :  access (33)
Subject: Compartments :  none
Permissions :  none
Class :  0
Categories :  none
Access Mode :  read
Object: Level: 0 uid:user2(321) gid:group2(200) device:34,73 inode: 6157
Pathname :  /users/data/private
Compartments :  none
Class :  0
Categories :  none
Mode :  100551
```

In this example user1 tried to read the file /users/data/private which has permissions of 551. Since user1 did not have read access to the file a record was cut showing the data access violation.

UNICOS system log entries have a record format which is not convenient for processing and reporting. The system administrator may use the awk command to put all the key information on a single line, making it easy to use tools such as sed, grep and awk to create reports:

```
/etc/reduce -g $grp -f $logfile -t desc -p | awk '{

  /Discretionary/ {
       month=\$1
       day=\$2 }

  /e_ids/ {
       id=\$2 }

  /Access Mode/ {
       mode=\$4 }

  /Object/ {
       ownu=\$5
       owng=\$7 }

  /Pathname/ {
       path=\$3
       printf( "%s %s %s ATTEMPT %s TO %s OWN:%s,%s\n",\
               month, day, id, mode, path, owng, ownu }
}'
```

Producing a one line per record format:

```
Apr 21 user1 ATTEMPT read TO /users/data/file OWN:group2(200),user2(321)
```

Once each access violations has been condensed to one line it can be written to a file containing all access violations for the given time frame. A simple grep on the file for the phrase "`OWN:group2`" is all that is required to produce a data access violation report for group2.

**NFS ID Mapping**

NFS ID Mapping is a UNICOS tool which is easy to use and very helpful in maintaining data security in an NFS shop. The idea is to create maps between the users and groups on the Cray and the users and groups on particular workstation domains. This mapping ensures that all remote NFS users are who you think they are. There are no problems with duplicate user and group names allowing remote users to access data which they do not own. NFS ID mapping is also useful from an operational standpoint when user and group names are not consistent with in a network. Here is a brief overview of what is required to set up NFS ID mapping.

1. A file must exist in /etc/uidmaps/users called *domain*.passwd for each domain where user IDs will be mapped to Cray user IDs. See the script /etc/uidmaps/Get.domains for examples of automated ways to get the remote /etc/passwd files. There is no requirement that these passwd files be authentic. If the user IDs and user names on the remote host are known, a passwd file may be created.
2. A file must exist in /etc/uidmaps/groups called *domain*.group for each domain with group IDs mapped to Cray Group IDs. See the script /etc/uidmaps/Get.domains for examples of automated ways to get the remote /etc/group files. There is no requirement that these group files be authentic. If the group IDs and member names on the remote host are known, a group file may be created.
3. Exception files are created to map names and groups for each domain. The exception files are only necessary when a user or a group has a different name on the Cray and in the workstation domain. The exception files tell the mapping code who is who. See the man page for the nfsmerge command for details on creating exception files.
4. Create a list of hosts in each domain which will be using NFS. Call these files /etc/uidmaps/hosts/d*omain* for ease in writing a script to handle the administration.
5. Create a list of domains to be mapped and use the nfsmerge command to create group and user ID maps for each domain.
6. Create a list of hosts in each domain and use the nfsaddhost and the nfsaddmap commands to load the maps into the kernel to provide NFS ID mapping for the specified hosts.

Creating the exception files is largely a manual process and the most time consuming portion of the NFS ID Mapping implementation. The exception files are required to show where to map each inconsistent user and group. This process will be much faster for installations that have consistent user and group names across all UNIX platforms.

To make administration of NFS ID mapping easier, put all nfsmerge and nfsaddmap commands into scripts to run whenever the mapping needs update. A loop can be used to issue the lengthy nfsmerge command for each domain:

```
USRS=/etc/uidmaps/users
GRPS=/etc/uidmaps/groups
LOGS=/etc/uidmaps/log
MAPS=/etc/uidmaps/maps
HOST=/etc/uidmaps/hosts
DOMAINS=`cat /etc/uidmaps/domains`
for domain in $DOMAINS
  do
  mv $MAPS/u.cray.$domain $MAPS/u.cray.$domain.old
  mv $MAPS/g.cray.$domain $MAPS/g.cray.$domain.old
  /etc/uidmaps/nfsmerge       -L $USRS/e.cray.$domain \            #UID Exceptions       input
                              -u $MAPS/u.cray.$domain \            #UID Map              output
                              -E $GRPS/e.cray.$domain \            #GID Exceptions       input
                              -g $MAPS/g.cray.$domain \            #GID Map              output
                              /etc/passwd $USRS/$domain.passwd \   #passwd files         input
                              /etc/group  $GRPS/$domain.group  \   #group files          input
                              > $LOGS/l.cray.$domain               #audit trail          output
done
```

A script may also be used to install the maps into the syetm kernel:

```
/etc/uidmaps/nfsidmap -d                        # Disable ID mapping
/etc/uidmaps/nfsclear                           # Clear kernel NFS Map table
/etc/uidmaps/nfsaddhost -l localhost            # Add loopback entry
for domain in $DOMAINS
  do /etc/uidmaps/nfsaddmap -u $MAPS/u.cray.$domain -g $MAPS/g.cray.$domain  $domain
  for host in $HOST/$domain
    do /etc/uidmaps/nfsaddhost -d $domain -c -s -l $host
done ; done
/etc/uidmaps/nfsidmap -e                        #enable ID Mapping
```

## Managing Users

### Login Ownership

It is important that a system administrator knows who has access to the system and can audit the activities of each individual. To accomplish this there must be a one to one correspondence between user names and humans. If it is necessary to share a user name among several people, each person should log into the system with their own user name and use the switch user command to access the shared ID.

Each user name should be mapped to an owning organization. The owning organization is ultimately responsible for the actions of their users. Since all groups are mapped to a data owner, it makes sense to map each user name to one group and let the data owner act as ID owner as well. A simple way to do this is to let the first group in a user's GID field. the "primary" group, map to his organization. In user communities with frequent personnel changes it may be difficult to keep primary groups up to date. An automated system with links into the payroll data is a good way to keep track of people because most business are very good about keeping payroll data current. A nightly scan would detect transferred and terminated employees. Logins for transferred users should be suspended until the new organization agrees to take responsibility for the user. Logins must be suspended or removed for employees who have left the company.

Adding a field to indicate primary groups allows an easy way to create reports which look at people on the system:

```
#group:     owner:      organization:     mailstop:         primary?
research:   Jane Doe:   R&D:              123 Bldg. A:      1:
testing:    John Smith: Systems:          999 Bldg. Q:      0:
production: Jill Smith: Systems:          111 Bldg Q:       1:
operations: A. Jones:   Operations:       100 Bldg. Z:      0:
```

A awk command may be used to create list of primary groups in a given organization. Let $grpown be set to the group ownership file:

```
getpri ()

{ cat $grpown |\
  awk -F: -v org=$1 '{
    if ($2==org)
     {if ($5 == 1)
       {prigrp[i]=$1
        i++}}}
  END{ for ( i in prigrp ) print prigrp[i] }' }
```

Now the report for each organization is structured as follows:

```
for org in `cut -d: -f3 $grp.own | sort -u`
  do for group in `getgrps $org`
     do
     processing for each group: data issues
   done
   for primary in `getpri $org`
     do
     processing for members of the primary group: user issues
   done
done > $organization.rpt
```

### Login Aging

Users rarely notify administrators when they cease to use a system. It is wise for the administrator to implement a system to "age" unused IDs. After an initial threshold for lack of use is reached a login is suspended. Once a second threshold is reached a login is removed from the system.

Various UNICOS utilities show different dates for last system usage. Discrepancies in dates come from the fact that some tools record just the last interactive login while others consider the use of NQS batch, switch user, and background processes as access. For the sake of login ID aging, it is important to look at the last time the ID was used in any way, shape or form. The most accurate place to collect this data would be from the UNICOS security logs. However, the overhead of searching logs to determine the last use for each user is not practical. The next best place to collect this data is in the UNICOS accounting system with the file /usr/adm/acct/sum/loginlog.

The loginlog file tracks a date next to each user name. If there is no record of system usage the date is recorded as 00-00-00 and it is necessary to age these logins as a special case. One solution would be to let the code which adds login

IDs cut a record to a file with the data of installation. If 00-00-00 is encountered in the loginlog, the aging routine would go to the this file and determine account age based on the date the login was added to the system.

There are also system logins such as bin, sys, Idle, sync which may never be used but are left on the system as users of data. This special case may be handled by maintaining a list of exempt logins in the aging routine.

UNICOS does not currently support login expiration dates but other systems at an installation may have this capability. It may make sense to suspend logins which have expired elsewhere.

The Login Aging Routine may report the following information to each organization monthly:

- Logins which will be removed if not used within a month
- Logins which have been removed due to lack of use
- Logins which have never been used and are aging based on date of installation
- Logins which will be suspended if not used within a month
- Logins which have been suspended due to lack of use
- Logins suspended due to expiration date on other systems
- Logins which active and enabled

The UNICOS User Database (UDB) includes field called "disabled" which may be set to one to suspend a login. When a login is removed via the UNICOS "New User" routine it removes the user's home directory but leaves data owned by deleted ID in other parts of the system. Each installation should work with the data owners to determine a way to manage data disposition when a user is removed from the system. It is essential that any code which removes or changes a home directory do various checks to ensure it never executes on the home directory of /.

**Satisfying User Requests**

User support becomes an important topic as the size of a system increases. While it is desirable to limit the number of people with full root capabilities, more support staff is required to satisfy user requests. Problems occur in identifying users and obtaining authorization to perform requested actions. Audit trails of specific actions taken by root are difficult to reconstruct, making it difficult to provide reports for data owners. Another problem which may occur when multiple root users are handling user requests is inconsistent results which leads to frustration and wasted cycles.

These problems can be eliminated by use of an application to handle user requests. To create the application, the root users can put commands used into a script each time a user request is satisfied. These scripts may then be invoked from a menu which may restricts the administrator to operating on IDs in particular organizations. Commands which require privileged may be invoked SUID to root. A routine to log all actions may be added to make reporting administrative actions easy. The application will allow the distribution of user support to administrators in the user community. The following user support functions may be included in the application:

- ID Maintenance
  - Add New Login
  - Remove User Login
  - Suspend User Login
  - Unsuspend User Login
  - Change a users shell
  - Change a users tape limits
  - Change a users login name
  - Change a users user ID number
  - Change a users password
  - Run cll to reset password fail counter

- Group Maintenance
  - Add a New Group
  - Remove A Group
  - Add User to Existing Group
  - Remove a User from a Group
  - Change a Users Primary Group
  - Show Group Owner and Members

- Resource/Privilege Changes
  - Add/Remove MPP access
  - Remove ALL privileges from a user
  - Remove ALL admin groups from a user
  - Add/Remove a charge code for a user
  - Add Operator privilege to a user
  - Add Admin account for a security contact

Change users tape resource limits
Change users data migration threshold

- Data Movement and Access
  Move data
  Copy data
  Delete data
  Restore data from backup
  Create a directory
  Change ownership
  Change group
  Change access mode
  Change account code

- Information
  Display user information
  Display user UDB record
  Scan system logs for user login failures
  Show group owner and membership
  Long listing of directory contents

A paper with details about implementing such as system was presented at the Spring 1995 Cray User Group Meeting. See the Spring 1995 *Proceedings of the Cray User Group*, p349-354, "Distributed User Support on Exxon's Cray Supercomputer". The paper is also available on the Internet from the Security MIG's page off the CUG home page: http:\\www.cug.org.

**User Violation Reporting**

In UNIX users have the ability to do things that can compromise system security. It is wise to monitor certain aspects of user activity to ensure that the users do not leave the system in a vulnerable state. Some suggested areas for monitoring user activity:

- File permissions: To reduce the risk of Trojan horse attacks, do not allow users to give any one write access to his home directory or environmental files.

- Trusted hosts: Do not allow users to share accounts via the trusted host mechanism. If possible, don't allow users to use the trusted hosts mechanism at all. If you must use these functions, the .rhosts files kept by users need to be monitored. These files should not contain wild cards, should only specify logins that belong to the person who uses the account and file permissions on the .rhosts file should be set to 600.

- FTP Access: A .netrc file may be kept to define user, machine, passwords for FTP. It is necessary to scan the contents of .netrc files to ensure the users do not have hard coded passwords. FTP cannot be used by a batch job to transfer files without a hard coded password in the .netrc file. Another method of file transfer will be required.

- Switch User: The sulog in /usr/adm will show when users switch user to each other. If two people are working together then the use of switch user is valid and even desirable. However, the use of switch user may also indicate password and account sharing.

- Passwords: Although the encrypted password field is not visible in UNICOS, encrypted passwords may be visible on other systems. Users frequently use one password throughout the network. If a password is breached on one machine, it may be breached all over the network. Educate users to use different passwords on each platform. It is sufficient to vary a single character from machine to machine. Educate administrators to set good passwords on behalf of users. It may be worth while to run Crack against the encrypted passwords and notify users if their password is breached.

- Failed login attempts: A administrator should be aware of the "typical" state of failed password counts on his machine and investigate if this number suddenly increases. The UNICOS `/etc/cll -L` command will show all non-zero failed password counters on the system. If the UNICOS system log is configured to record all failed login attempts, the administrator could gather details on how a given user managed to fail out his login. Note that the failed password counter is increased from zero to the disable threshold in one fell swoop when a user fails the /bin/su command three times in a 60 second interval.

- Inactive sessions: People leave terminals unattended, no matter how often they are reminded of the importance of logging out and locking screens. It is wise to create a cron script to periodically looks for inactive sessions and logs them off. Each installation will have different thresholds to determine an "abandoned" session, be careful not to terminate long running batch jobs accidentally. The routine to perform inactive session log out should be tested in logging mode for a few days before going production.

- The spcheck command: The UNICOS MLS System comes equipped with a command that will scan the system and perform many security checks, including:
  - List users who have administrative categories.
  - List all setuid and setgid files in the /bin, /usr/bin, /usr/lib, and /etc directories.
  - Report users in groups root, adm, bin, and sys.
  - Report infrequently used login IDs
  - List .profile and .cshrc files that are writable by anyone.
  - Report users with duplicate user IDs
  - List users who cannot change their passwords
  - List users with passwords that do not expire
  - Report users with numerous switch user failures
  - Report files that have the world read or write permission set
  - Report programs with the setuid or setgid bits set.
  - Report block/character special files not in /dev.
  - Checks /etc/passwd and /etc/group files for consistency

## Managing Privilege

Although there are tools available to segregate all or part of Superuser privilege, this paper will only look at control measures for a system running the Superuser privilege policy. In native UNIX systems, all privilege comes from the Superuser root account.

Frequently people become confused when discussing privilege on UNIX systems. Issues such as system installed login IDs and groups that access proprietary data cloud this issue. Privilege in a native UNIX system is really a very simple concept, all privilege comes from the superuser account, root. System IDs are simply another account. Membership in a particular group controls access to data. While a group may control access to SUID to root programs, the privilege still comes from the fact that the program runs as root.

Superuser must be trusted because it is all powerful, with the ability to change audit logs, reconfigure the system and access all user data. Control around the use of the all powerful root account is very important.

To make the task of placing controls around root usage manageable, it is useful to look at each way a process could become the superuser. There are only five ways to obtain root privilege:

1) The system is in single user mode
2) A person with the root password logs directly into the root account
3) A person with the root password uses the switch user mechanism to become root
4) An SUID to root program is executed
5) A program is executed from root's crontab

Placing reasonable control measures around each of these ways to reach root goes a long way toward obtaining control over the elusive superuser account.

### Single User Mode

Single user mode is the state the system is in when it is down for maintenance. The first control to manage single user mode is to make sure the system is completely down. TCP/IP should not be running, users should not be able to log in. The batch subsystem should be disabled and unnecessary file systems should not be mounted. Minimizing system activity will reduce the risk of unauthorized access and changes while in single user mode.

All changes should be planned, documented and approved by management before the system is bought down for maintenance. The administrators should create a detailed plan of commands to execute during down time and this plan should be approved by several technical people.

The UNICOS slog daemon will run in single user mode and collect security log data. However, it is likely that the problems associated with storing this data during single user mode will make it impractical to collect log data while file systems are offline. An installation may opt to turn off security logs in single user mode to prevent buffer overflow. When security logs are not active it is extremely important to have multiple people present during down time. The spaudit command which enables system logging should be added to the automated startup processes to ensure that security logging is enabled when the system is returned to multi-user mode.

### Direct Login to root

Its not wise to allow system administrators to log directly into the root account because there is no audit trail to indicate which individual accessed root. The UNICOS system logs may be scanned with */etc/reduce -u root -t logn* to ensure no direct root logins have occurred via any mechanism. The administrators should not access root via telnet, rlogin, rcp, rsh, ftp, NQS, etc, etc, etc. People should log in as themselves then use bin/su to access root, providing a audit trail of the access.

**Switch User to root**

This is the preferred method for accessing the root account in multi-user mode. Configure the slog daemon to record all occurrences of su to root because the /usr/adm/sulog audit trail is in lain text and easy to alter. In the event that specific audit trails of root actions are required, the UNICOS accounting system records all commands issued by a given login session.

Limit the number of people with the root password. This can be done by using other privilege mechanisms to allow access to commands in administrative command. SUID to root code, UDB privileges and Privilege Access Lists are all mechanisms that may be used in a UNICOS system to limit the number of root users.

Encourage root users to minimize the time spent in root, reducing the risk of mistakes and making it easier to audit root activity. A good way to help root users remember to perform just one function then exit root is to require a comment whenever a su to root is performed. A front end to the /bin/su program may query the user for a reason for the root access. The data collected is useful for management review and to recognize when a root user may be able to do his job with out access to root.

**SUID to root Code**

Many of the SUID to root programs that are a part of the UNICOS system are intended for use by the entire user community. These programs run as root to perform specific privileged action on behalf of a user. Other SUID to root programs are restrict to a particular group, creating the issue of "privileged groups" on the system. This is not a problem as long as the administrators and data owners are aware that membership in particular groups allows the user to execute programs as root. It may be preferable to keep a clear distinction between groups and privilege. Let groups control data access and use Access Control Lists (ACLs) to define who can execute restricted SUID to root code. There are several benefits to this scheme, including a clear distinction between data access and privilege assignment, easy review of privileged users, and no chance of accidental privilege acquisition when someone is added to a group.

Many installations create local SUID to root code to help limit the number of people with the root password or allow the user community to run a privileged command. This is a good control measure provided detailed guidelines are in place to ensure that local SUID to root code is properly protected and authorized. It is important to treat the installation of local SUID to root code as an assignment of privilege. Multiple administrators should check the code to ensure that proper traps are in place to keep users from issuing unintended commands as root. Reasonable file protections should be placed on the code and an ACL may be in order to limit access. All use of local SUID to root code should be properly documented and approved by system and data owners.

**Programs Executed from roots crontab:**

The UNIX scheduler, cron, runs programs on behalf of users. Each authorized user may create a table, crontab, which lists programs to be run by cron at a specific time and day. It is important to protect programs which are called by root's crontab. Controls should be put in place to ensure that only root users can write to these critical files and the directories in which they reside. Consider using a library to store all programs called by root's crontab for ease in monitoring contents and access. Only root users should update these programs. All changes need to be properly authorized and tested before production. Programs run by root's crontab must be treated as high risk; they kick off unattended and run as root, making them vulnerable to Trojan Horses and malicious code.

## Summary

The topic of control measures which may be used to increase system security is huge, this paper barely scratches the surface. Hopefully administrators of large UNIX systems got ideas for easy ways to manage their system to increase system security. Numerous books are available on UNIX security and huge amounts of data on the topic can be found on the Internet. Anyone charged with the task of securing a UNIX system is strongly encouraged to continue this research and share their finding with the rest of the computing community. UNIX security has improved by leaps and bounds in the past few years and this trend is bound to continue as long as UNIX remains a key part of business and system administrators are diligent in minimizing risks in their computing environment.