Administering the System Audit Trail

The System Audit Trail features allow administrators to review a record of all system activity. The ongoing record of system activity shows general trends in system usage and also violations of your system use policy. For example, any unsuccessful attempts to use system resources can be recorded in the audit trail. If a user consistently attempts to access files owned by other users, or attempts to guess the root password, this can be recorded also. The site administrators can monitor all system activity through the audit trail. Sections of this chapter include:

Note that references are made in this chapter to auditable "MAC" and "Mandatory Access Control" events, such as an event generated when an attempt is made to access a file protected by a higher MAC clearance. The audit system provides facilities to audit all events on all IRIX operating systems. Mandatory Access Control (MAC) is available only in the Trusted IRIX/B optional operating system. No MAC audit events are generated by standard IRIX. If you have installed Trusted IRIX/B, you will have received additional documentation describing the special security features in that product. Users of standard IRIX can safely ignore all references to MAC, labels, and the *dbedit*, *chlabel* and *newlabel* commands. To find out if your system is running Trusted IRIX/B, use the *uname* command with the **-a** option. Standard IRIX systems give a response that looks like this:

`IRIX` *System_name* `5.1 02131441 IP12`

If your machine is running Trusted IRIX/B, the name IRIX in the above example will be replaced with "Trusted IRIX/B."

Discretionary Access Control (DAC) is the term used by the auditing subsystem for the standard UNIX system of file permissions. IRIX uses the standard permissions system common to all UNIX based operating systems.

Enabling Auditing

The audit subsystem is distributed with your IRIX operating system media, but is not installed by default. To enable auditing, you must use Inst to install the *eoe2.sw.audit* software package from your distribution media. *inst* is described in detail in the *Software Installation Guide*. Once this package has been installed, reboot your system and use the *chkconfig* utility to enable auditing. The chkconfig(1M) reference page provides complete information on the use of *chkconfig* but, simply described, you will see a list of configurable options and a notation of *off* or *on* for each option. The list is in alphabetical order.

For example, here is a partial *chkconfig* listing that includes the audit option:

```
Flag                    State
====                    =====
audit                   off
automount               on
windowsystem            on
xdm                     off
```

The following command enables auditing on your system:

```
chkconfig audit on
```

The system immediately begins collecting audit data on the default set of audit events. The default audit events are listed and described below.

Default Auditing

The default auditing environment is already set up when you install IRIX. You need not take any action to maintain the default auditing environment. Within your default IRIX distribution, there is a file called */etc/init.d/audit*. This file contains the default audit trail initialization. The default auditing selections produce a full record of system activity with a minimum of disk-space usage. The following list contains all event types audited by default. (The individual event types are not described in this list, but a description for all event types is given in “Auditable Events” on page 137.)

sat_access_denied	sat_chdir	sat_chroot
sat_open	sat_file_crt_del	sat_file_crt_del2
sat_file_write	sat_mount	sat_file_attr_write
sat_exec sat_sysacct	sat_fchdir	sat_tty_setlabel
sat_fd_attr_write	sat_proc_read	sat_proc_write
sat_proc_attr_write	sat_fork sat_exit	sat_proc_attr_write
sat_proc_attr_write2	sat_svipc_create	sat_svipc_remove
sat_svipc_change	sat_bsdipc_create	sat_bsdipc_create_pair
sat_bsdipc_shutdown	sat_bsdipc_mac_change	sat_bsdipc_expl_addr
sat_hostid_set	sat_clock_set	sat_hostname_set
sat_domainname_set	sat_ae_custom	sat_ae_identity
sat_ae_dbedit	sat_ae_mount	

Customizing Auditing

When you have installed your system, you can select the level and type of auditing that you wish to use. The default auditing environment described above is created for you at installation time. For most purposes this auditing environment is satisfactory. However, remember that the System Audit Trail is completely configurable at any time through the *sat_select* and *satconfig* utilities. The *satconfig* utility is the preferred tool for use on graphics systems, since it provides a convenient graphical interface for switching each auditable event type on or off. The *sat_select* command is useful for server users and others who do not wish to use the *satconfig* utility. These utilities are discussed in detail in “Using satconfig” on page 141 and “Using sat_select” on page 142.

What Should I Audit?

You can audit all system activity or certain types of activity, such as file removal or access denial. Users are tracked through the audit trail by User ID (UID) numbers. Any audited activity is associated with the UID of the person who performed that action. It is a central feature of the System Audit Trail that though the effective UID changes with the use of the *su* command, the SAT ID does not. All of a user's actions after logging in are audited at the original login UID.

When you select the type of activities to audit, there are still several options for auditing. For example, if you wish to monitor the removal of files, you can generate an audit record under two conditions:

- when the action fails (sat_access_denied, sat_access_failed)
- when the action succeeds (sat_file_crt_del, sat_file_crt_del2)

Many different types of activities take place on your trusted computer system. There are login attempts, file manipulations, use of devices (such as printers and tape drives), and administrative activity. Within this list of general activities, you may choose to audit many specific kinds of actions.

Below is a list of auditable actions with a short definition of each action and one or more of the appropriate event types that can be audited. Important actions contain a note that they should always be audited:

- login and logout (sat_ae_identity)

  Any login attempt, whether successful or not, should be audited. Also, an audit record should be generated when the user logs out of the system.

- su (sat_check_priv, sat_ae_identity)

  Whenever a user invokes the *su* command, whether to super-use some administrative account, such as root or another user account, the event should be audited. This is especially true for unsuccessful attempts, as they may indicate attempts at unauthorized access.

- chlabel and newlabel (file_attr_write, sat_proc_own_attr_write)

  Any time a user changes a MAC label on a Trusted IRIX/B system, it is wise to make an audit record of the event. (This does not happen under standard IRIX.)

- password change (sat_ae_identity)

  Whenever a user changes his or her password, it is wise to make an audit record of the event.

- administrative activity (sat_ae_mount, sat_clock_set, sat_hostid_set, etc)

  Any activity related to system administration should be carefully audited; for example, editing the */etc/fstab* file.

- DAC permissions change (sat_fd_attr_write, sat_file_attr_write)

  When a user invokes the *chmod* command to change the DAC permissions on a file or the *chown* command to change the ownership of a file.

- file creation (sat_file_crt_del, sat_file_crt_del2)

  Whenever a new link, file, or directory is created.

- file deletion (sat_file_crt_del, sat_file_crt_del2)

  Whenever a link, file, or directory is removed.

- process activity (sat_exec, sat_exit, sat_fork)

  When a new process is created, forked, exited, or killed.

  The audit administrator (auditor) can change the audited events by entering a new *sat_select* command. It is possible to change the selected event types at different times of day, by using the *cron* utility to execute *sat_select* periodically.

  To tailor your auditing for your specific needs, use the *sat_select* or *satconfig* utilities.

Auditable Events

The following is a complete list of auditable event types:

sat_access_denied
: Access to the file or some element of the path was denied due to enforcement of MAC or DAC permissions.

sat_access_failed
: Access to a file was denied because the path specified does not exist.

sat_chdir
: Current working directory was changed with *chdir*.

sat_chroot
: Current root directory was changed with *chroot*.

sat_open
: A file was opened with write permission.

sat_open_ro
: A file was opened read-only.

sat_read_symlink
: The contents of a symbolic link were read with *readlink*. Note that the file the link "points" to is not accessed in any way.

sat_file_crt_del
: A file was added or removed from a directory.

sat_file_crt_del2
: This is the same as sat_file_crt_del, but reports that two files (perhaps a link) were removed.

sat_file_write
: The data in a file was modified by *truncate*.

sat_mount
: A filesystem was mounted or unmounted.

sat_file_attr_read
: The attributes of a file were read by *stat*.

sat_file_attr_write
: The attributes of a file were written by *chmod*.

sat_exec
: A new process has been introduced by *exec*.

sat_sysacct
: System accounting has been turned on or off.

sat_fchdir
: The user changed from the current working directory to the directory "pointed" to by the given open descriptor.

sat_fd_read
: Information was read from a file descriptor using *read*.

sat_fd_read2
: The same event as sat_fd_read, but with multiple file descriptors.

sat_tty_setlabel
: The user set the label of a port via *ioctl*.

sat_fd_write
: The user finalized a change to a file descriptor.

sat_fd_attr_write
: The user changed the attributes of the file "pointed" to by the given file descriptor using *fchmod*.

sat_pipe
: The user created an unnamed pipe.

sat_dup
: The user duplicated a file descriptor.

sat_close
: The user closed a file descriptor.

sat_proc_read
: The user read from a process's address space using *ptrace*.

sat_proc_write
: The user finalized a changes to a process's address space using *ptrace*.

sat_proc_attr_read
: The user read a process's attributes.

sat_proc_attr_write
: The user finalized a change to a process's attributes.

sat_fork
: The user duplicated the current process (thereby creating a new process).

sat_exit
: The user ended the current process.

sat_proc_own_attr_write
: Process attributes were changed.

sat_clock_set
: The system clock was set.

sat_hostname_set
: The hostname was set.

sat_domainname_set
: The domain name was set.

sat_hostid_set
: The host ID was set.

sat_check_priv
: Action requiring superuser privilege was performed.

sat_control
: The *sat_select* command was used.

sat_svipc_access
: The user accessed a System V IPC data structure.

sat_svipc_create
: The user created a System V IPC data structure.

sat_svipc_remove
: The user removed a System V IPC data structure.

sat_svipc_change
: The user set some attribute of a System V IPC data structure.

sat_bsdipc_create
: The user created a socket.

sat_bsdipc_create_pair
: The user created a socket pair.

sat_bsdipc_shutdown
: The user shut down a socket.

sat_bsdipc_mac_change
: The user changed the MAC label on a socket.

sat_bsdipc_address
A network address was used explicitly via the *accept*, *bind*, or *connect* system calls.

sat_bsdipc_resvport
A reserved port was successfully bound.

sat_bsdipc_deliver
A packet was delivered to a socket.

sat_bsdipc_cantfind
A packet was not delivered because the socket could not be found.

sat_bsdipc_snoop_ok
A packet was delivered to a raw (snoop) socket.

sat_bsdipc_snoop_fail
A packet was not delivered to a raw socket because it was prevented by MAC policy.

sat_bsdipc_rx_ok
A packet was received on an interface.

sat_bsdipc_rx_range
A packet was not received, due to MAC violation outside the allowed label range on that interface.

sat_bsdipc_rx_missing
A packet was received on an interface with a missing or damaged MAC label.

sat_bsdipc_tx_ok
A packet was sent on the interface.

sat_bsdipc_tx_range
A packet was not sent, due to a MAC violation.

sat_bsdipc_tx_toobig
A packet was not sent, because the MAC label was too large for the IP header to contain.

sat_bsdipc_if_config
An interface structure's attributes were changed.

sat_bsdipc_if_invalid
Attempt to change MAC labels was disallowed for lack of MAC privilege.

sat_bsdipc_if_setlabel
: The MAC labels on an interface structure were changed.

All *sat_ae* events are used for application auditing, which means that a privileged program generated the record, rather than the kernel.

sat_ae_identity
: A login- or logout- related event occurred.

sat_ae_dbedit
: A file was modified using the *dbedit* utility. (This utility is available only with the Trusted IRIX/B optional product.)

sat_ae_mount
: An NFS filesystem was mounted.

sat_ae_custom
: An application-defined event occurred. Application developers can engineer their applications to generate this event.

Using satconfig

satconfig is a graphical utility that you use to configure exactly which events will be audited on your system. Any user can invoke *satconfig*, but only the superuser may actually change the auditing environment. When you invoke *satconfig*, a new window opens on your screen. The main body of the window has a list of all the available event types. Next to each event type name is a button. At any time, each button is either up or down. If the button is down, the event type is selected for auditing. If the button is up, the event type is not audited. Use your mouse and the left mouse button to select whether you want the event type in question to be on or off.

When you first begin using the audit trail, there is a default set of audited events. You can modify that selection using *satconfig*, but the *satconfig* window contains a pulldown menu labeled "edit" that you can use at any time to set the auditing environment to a few preset environments. These include the original SGI default audit selections, your local default selections, all event types selected, no event types selected, and a current events selection. The current events selection restores the auditing environment that was last saved on your machine. The local default environment can be any combination of event types that you choose. You create a local default environment by following the instructions in "Saving and Retrieving Your Auditing Environment" on page 143.

At the bottom of the *satconfig* screen there are three buttons. These buttons are labeled *Apply*, *Revert*, and *Quit*. When you have made your auditing selections, use the left mouse button to press the *Apply* button on the screen to activate the auditing selections. If you change your mind while making audit selections, you can use the *Revert* button to reset the individual event type buttons to the selections currently in use. The third button is labeled *Quit* and closes the *satconfig* window. If you have made selections that have not been applied, *satconfig* asks you if you really want to quit and discard the changes you have made without applying them.

Using sat_select

The *sat_select* utility is a character-based program that modifies your audit event type selections. Additionally, you can use the *sat_select* utility to change your local default auditing environment or to read in a preselected set of event type choices from a file. In this way, you can have several preset auditing environments ready in files for various situations and switch between them conveniently. If you have a graphical system, *satconfig* is the suggested utility for administering your auditing event type selections. *sat_select* exists for non-graphics systems and for making large-scale, file-oriented changes.

For complete information on using *sat_select*, consult the sat_select(1M) reference page, but in general, the syntax most often used is

sat_select -on event

and

sat_select -off event

sat_select -on event directs the system audit trail to collect records describing the given event. If "all" is given as the *event* string, all event types are collected.

sat_select -off event directs the system to stop collecting information on that event type. If "all" is given as the *event* string, all event types are ignored.

sat_select issued with no arguments lists the audit events currently being collected. The effect of subsequent *sat_select* programs is cumulative. Help is available through the **-h** option.

Saving and Retrieving Your Auditing Environment

From time to time you may wish to change your auditing environment. You do this with the *sat_select* command. If you are making a temporary change, you may wish to save your current auditing environment for easy replacement. To do so, use this command:

```
sat_select –out > /etc/config/sat_select.options
```

Then, to restore auditing to the saved state, use this command:

```
sat_select 'cat /etc/config/sat_select.options'
```

The single quotation marks in the above example are crucial and must not be omitted.

You may save as many different audit states as you wish, in different filenames. Simply insert the filename of the state you wish to use in the above example. The */etc/config/sat_select.options* file is the default audit state file that is read at boot time. The */etc/config/sat_select.options* file must be labeled *dblow* if you are running Trusted IRIX/B, and you should restrict DAC file permissions to root only regardless of your operating system type.

Placing the Audit Files

The location of your audit record files is also configurable. You can direct your audit records to be saved to any location you desire, including magnetic tape. *satd* saves its input data in the directories or files named in its *path* arguments.

The **-f** option to *satd* specifies an output path, which may be a directory or a file. If the output path is a directory, *satd* creates and fills uniquely named files under that directory. (Files are named for the time of their creation. For instance, file *sat_9101231636* was created in 1991, on January 23, at 4:36 pm.) If the output path is a specific filename, *satd* writes to that file.

You can specify several output paths in the *satd* command line. To do so, you must precede each path with a **-f** or put commas (but no blank space) between each pathname. Taken together, all of the output paths specified in the command line are known as the path list. Here are a pair of examples of command lines that contain path lists:

```
satd –f /sat1 –f /sat2 –f /sat3 –f /dev/null

satd –f /sat1,/sat2,/sat3,/dev/null
```

If no output paths are specified after the **-f** flag, the audit trail records are not saved anywhere, and the system halts. If a path given as a command-line parameter is invalid for any reason, a warning is printed, that path is omitted from the path list, and *satd* continues operating with whatever specified paths are valid. If the specified path does not already exist, *satd* creates a file with that name.

A file or directory is full when the filesystem on which it resides has no more available space. If a directory is specified as an output path, an audit file is constructed under that directory. When the audit file is filled to an internally specified maximum size, it is closed and a new audit file is created under that directory.

When one output path becomes full, *satd* replaces the current output path with a path that is not full. The method of replacement is configurable with the **-r** option. The output path is also replaced if *satd* receives a SIGHUP signal, for instance one sent with a *kill* command.

If an output path becomes nearly full, warnings are displayed to the system console to notify the administrator to move the audit trail to tape. If all of the output paths become completely full, the system state moves to single-user mode with a very short grace period.

In order to protect against the loss of data due to sudden system state changes, when satd begins operations, it creates a file called */satd.reserve*, which is exactly 250,000 bytes long. If *satd* runs out of space, it immediately removes the *satd.reserve* file to free the 250,000 bytes for use to store audit records while the system moves to single-user mode. While the system is coming down, *satd* stores audit records in a series of files named */satd.reserve-n*, where *n* starts as 0. While *satd* is doing this, it issues a warning via *wall* to all users that they have ten seconds before system shutdown.

If the file */satd.emergency-0* already exists, *satd* immediately moves to the first available filename, typically */satd.emergency-1*. To guard against this happening, a warning is issued at boot time if any */satd.emergency* files exist.

For complete information on the audit daemon, see the audit(1M), satd(1M), and audit_filters(5) reference pages and the comments in */etc/init.d/audit*.

How to Audit a Specific User

At times, you may wish to examine the audit record of a particular user. For example, the user may have a history of violations of system security or may simply be leaving the project and an accounting of activity may be required.

If the user in question is being audited to determine if attempted security violations are taking place, use the command line:

sat_reduce -P *satfile* **|** **sat_summarize -u** *user_name*

This command line selects only the audit records that represent attempted violations. The **-P** flag to *sat_reduce* selects for attempted violations. The **-u** flag to the *sat_summarize* command lists the number of records generated by the user.

It is vitally important to remember that not every record of an attempted violation really represents malicious intent on the part of the user! Most of these records are generated in the course of normal work. The auditor should be looking for a trend, such as repeated attempts to access information unnecessary in the course of normal work (for example, a programmer attempting to access salary or hiring information).

In the second scenario, where the employee is leaving the project, the auditor is looking for a comprehensive list of files used by that employee so that the correct files and directories may be assigned a new owner who is remaining on the project.

The above listed command line provides a basic look at the user's activity. Next, to more closely examine the user's activities, issue the following command:

sat_reduce -u *user_name satfile* **|** **sat_interpret** **|** **more**

The *sat_reduce* command selects all of the audit records generated by the user. Then, the *sat_interpret* command puts the records into human readable form. The output of *sat_interpret* is very large. If it is impractical to direct this output to a file, you should direct the output to your screen and view it with a screen paging program such as *more*.

Using these two command lines, you should be able to view a user's activities and come to a reasonable knowledge of the types of actions the user is taking on the system. You can also generate a specific record, in human-readable form, of all security violations or files and resources accessed.

How to Audit a File

At times, you may wish to examine all audit records pertaining to an individual file. Perhaps some changes have been made to an important file and the user who made those changes must be identified. Or perhaps an accounting of all access to a sensitive file is needed. To obtain a record for each time the file was opened, you must first make certain that the audit daemon is recording *sat_open* and *sat_open_ro* events. Use the *sat_select* command to ensure that these events are logged. To search the audit log for these events, use the following command line:

sat_reduce -e sat_open -e sat_open_ro *satfile* **|**
sat_interpret | grep *filename*

How to Audit a Label Under Trusted IRIX/B

If you are using Trusted IRIX/B, your system supports Mandatory Access Control (MAC) labels on all files and processes. This section explains how to check the audit trail of a given security label.

If you are using standard IRIX, your system does not support MAC labels, and attempts to read the audit trail for events relating to such labels will be futile.

Since the number of configurable labels in Trusted IRIX/B is great enough for each project or portion of a project at your site to have its own label, you may sometimes need to audit a specific label to generate a record of activity on that label. Use the following command to generate a log of activity on a label:

sat_reduce -l *label satfile*

The above command chooses only audit records that pertain to the given label. The following command syntax allows you to select more than one label for your report:

sat_reduce -l *label* **-l** *label2 satfile*

Once you have obtained output from *sat_reduce*, use the other auditing utilities, such as *sat_interpret* or *sat_summarize*, to view it according to your needs.

Understanding the Audit Data

The audit trail for an active system with full auditing can be too large for a single person to read and understand, and the entries in the trail that alert you to trouble are small and rare. If you were to read the raw audit trail to find an instance of policy violation, it would be like trying to find a needle in a haystack. Therefore, several utilities exist to help you reduce and interpret the raw audit data. The *sat_reduce*, *sat_interpret*, and *sat_summarize* commands can be used to remove superfluous information and format the audit history in succinct packages. See the reference pages for these commands for specific information on their usage.

After your raw data has been reduced and interpreted, an individual record looks something like this:

```
Event type = sat_ae_identity
Outcome = Failure
Sequence number = 5
Time of event = Mon Mar 11 12:46:13.33 PST 1991
System call = syssgi,SGI_SATWRITE
Error status = 0 (No error)
SAT ID = anamaria
Identity event = LOGIN|-|/dev/ttyq4|anamaria|That user gave an invalid
label.
```

The *sat_summarize* command provides a short listing of what types of records are in the audit trail and how many there are of each type. It's a useful tool for scanning the records quickly and identifying trends in system usage or consistent problems.

Remember that file pathnames within audit records are not the same as those in common usage through the shell on your system. Since the audit record is an exact log for security purposes, many attributes of the pathname that are designed to be transparent in normal usage are explicit in the audit log. For example, the double slash (//) means a directory level crossing (ordinarily represented through the shell with a single slash (/)). A slash followed by an exclamation point (/!) indicates crossing a filesystem mount point. The slash and ampersand construction (/@) indicates that the path is following a symbolic link. If you are running Trusted IRIX/B, you may also see a slash followed by a right angle bracket (/>), which indicates that the directory level being crossed into is a multilevel directory. The *egrep* utility supports this notation, so it is possible to specify this form of pathname notation in regular expression searches. Below are two examples of audit record pathnames:

```
/usr/!orange2/@/fri//usr//src//lib//libmls//libmls.a
```

```
/usr/!tmp/>L_e//sat//sat_9012280805
```

The system places the audit data in files on your system. Each file begins with the starting date and time of the file, the machine name, and the host ID, and ends with the stopping date and time. If your system is interrupted (for example, by a power failure), the audit file being used at that time will have no ending entry. The audit daemon automatically closes a file when it reaches a certain manageable size and opens another. A new file is always started when the system is brought up. For information on these files and their format, see the satd(1M) reference page.

Potential Security Violations

The overwhelming majority of records in an audit trail are the result of the normal actions of users doing their jobs. No automated tool exists to locate records that signify the actions of abusers trying to violate system security. Nonetheless, an administrator can apply some general rules to detect abuse or violation of security policy. This list of tips is neither complete nor universal. Each administrator must customize the list to meet the particular needs of each site.

Use and Abuse by Outsiders

Intrusion by outsiders is among the most feared of abuses. Fortunately, this kind of abuse produces distinctive audit record patterns and is easily detected. Below, are descriptions of several different subcategories of outsider abuse that can be detected by the audit system. Note though, that these kinds of patterns can also be generated by an authorized user who makes a mistake or is misinformed. Patterns of this type are described below.

Attempts at Unauthorized Entry

All attempts at unauthorized entry generate audit records of the *sat_ae_identity* event type. (Use *sat_select*, *sat_reduce*, and *sat_interpret* to collect and view these records.) The interpreted output of these events contains a text string that describes the attempt at entry. Intruders from outside your organization have a much higher instance of failed login attempts than your authorized users.

Three interesting text strings reveal attempts at unauthorized entry:

- unsuccessful login attempt

- that user gave an invalid label
- could not set the label connection for device

Here is an example of an interpreted audit record of an unsuccessful login attempt:

```
Event type = sat_ae_identity
Outcome = Failure
Sequence number = 1
Time of event = Mon Mar 11 12:45:40.34 PST 1991
System call = syssgi,SGI_SATWRITE
Error status = 0 (No error)
SAT ID = anamaria
Identity event = LOGIN|-|/dev/ttyq4|guest|Unsuccessful login attempt.
```

System Usage at Unusual Hours or From Unusual Locations

Usage of your system outside of normal working hours or, if your system maintains physical security of terminals, from unusual locations, is a matter of interest. In most cases, the usage of the system is legitimate, but each instance certainly bears notation and examination. Many potential violations of security from outside your user community happen during nonpeak hours, and rarely from within your physical site. To observe activity at odd hours, enter the following commands in order:

1. **sat_reduce -a** *start_time* **satfile > /usr/tmp/early+late**
2. **sat_reduce -A** *end_time* **satfile >> /usr/tmp/early+late**
3. **sat_reduce -U root -U sys -U daemon -U adm -U lp /usr/tmp/early+late > /usr/tmp/e+l_ordusers**
4. **sat_interpret /usr/tmp/e+l_ordusers | more**

If your site assigns a terminal to each user and maintains reasonable physical security for each terminal, you can monitor logins from unusual locations. For example, if a user normally working in a group computer lab makes a login attempt from a private office, this event may be cause for interest. To get a list of login events, enter the following command:

sat_reduce -e sat_ae_identity sat_file | sat_interpret | grep LOGIN

Bear in mind that it does not necessarily represent a violation of security if a user is working at an unusual terminal or even if a user is logged on at two or more terminals at once. For instance, the user may be correcting a mistake and may have logged in elsewhere explicitly for the purpose of terminating unwanted processes. You should be looking for instances where the user is not genuinely logged in twice, but where one instance of the login is an intruder.

Connections with Machines Outside the Local Network

Whenever a user connects to a machine outside your trusted local network, an audit record should be generated. A connection to a host outside of the local network is worthy of notice but not necessarily a violation of security. You should be on the lookout for trojan horse programs that cause your system to make the outward connection at a later time. You can identify outward network connections with the following command sequence:

1. **`sat_reduce -e sat_bsdipc_addr`** *satfile* **`>`** */usr/tmp/connect*
2. **`sat_interpret`** */usr/tmp/connect* **`>`** */usr/tmp/connect.int*
3. **`grep -n "Remote host"`** */usr/tmp/connect.int*

The above command sequence is dependent on the specific implementation of your networking software. You may need to modify your command line to reflect your networking situation. For example, if the software you are using does not generate the *sat_bsdipc_addr* auditing event type, you should search for another event type that is generated.

Use and Abuse by Insiders

Beyond use and abuse by intruders, unfortunately, the possibility arises of abuse from within your organization. The following types of events are the most common instances of security violations. It is extremely counterproductive to assume that a security violation on the part of an authorized user indicates that the user is not trustworthy or is involved in some attempt to break security for malicious purposes. Most violations of system security by users involve a failure on the part of the Administrator to adequately prepare the working environment. Users are most concerned with accomplishing their work tasks, not with fixing the computer system to provide themselves with the correct tools. Therefore, you should not be suspicious of the user who violates security unless a clear pattern of a specific and unnecessary security violation is apparent.

File Permission Violations

Although the system records each instance where access to a file or resource is denied, the information contained in these audit records is rarely indicative of a security violation. Many applications and utilities operate on a principle of access denial as part of normal operation. These events are always logged, but only in rare cases do they indicate a violation. For example, the library function **getutent** always tries to open */etc/utmp* for read-write access. If this action fails, **getutent** immediately tries again, but requesting read-only access. Permissions on */etc/utmp* prohibit all users except root from opening this file for reading and writing. When an unprivileged user runs a program that calls *getutent()*, a *sat_access_denied* record is generated, and it is immediately followed in the audit trail by a *sat_open_ro* record, indicating that access was granted. The lesson in this example is that access denial is usually not indicative of a security violation.

The *sat_access_failed* event is often confused with the denial event. The event type is completely different and is even more rarely a cause for concern than access denial. When a user enters a command to an interactive shell (such as */bin/csh*), the shell tries to execute the command in each directory in the user's search path, failing at each attempt until it finds a directory that actually contains the command. Suppose a user enters *xterm* and his or her path variable contains

```
/bin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/bin/X11:~/bin
```

A *sat_access_failed* record is generated for each directory in the path until the command is found and executed. In this scenario, a record of failed access is generated for each of the following nonexistent programs: */bin/xterm*, */usr/bin/xterm*, */usr/sbin/xterm*, */usr/local/bin/xterm* and a successful *sat_file_exec* record for the real program: */usr/bin/X11/xterm*.

Unexpected Use of Root Privilege

Every interpreted audit record contains a line beginning with the keyword **Outcome**. The field following this keyword can be equal to one of **Success**, **Failure**, or **Success due to privilege**. The last case indicates that the user made a system call that would have failed except that superuser privilege was invoked to assure its successful completion. This is not necessarily a security violation or an unexpected use of root privilege. It is perfectly normal to see these outcomes. Any time an ordinary user runs a program that contains code that uses root privilege, **Success due to privilege** outcomes are generated. A good example of this kind of program is *passwd*. An ordinary user generates a record of this type simply by changing the password on his or her account.

What you should be looking for is an instance where the SAT ID or Effective ID field is different from the "User ID" field. This occurs when a user executes */bin/su* to gain root privileges or otherwise promotes the privilege level of a session. In most cases, this is not a security violation, since the root password is necessary to successfully complete the */bin/su* command.

An instance of using superuser privilege, though, is always worth examination in the audit trail. When you encounter an instance where a user has promoted his or her login session to root, you should check to see that the user is authorized to know the root password. If not, check whether the user indeed executed the */bin/su* command, or if he or she promoted the privilege of the session by some other means, such as a Trojan horse *setuid* shell command.

Whenever a user runs */bin/su* and thereby promotes the privilege of his or her login session, the auditor should also make a routine check of what actions the user took while the privilege was promoted.

Activity by Particularly Interesting Users

Sometimes a particular user is under official scrutiny by the management of a site. He or she may be on probation or may have just left employment under less than ideal circumstances. The auditor can choose to look at the records describing that user's behavior just by directing the audit trail through the *sat_reduce* command as follows:

1. **sat_reduce –u** *jeff* **< satfile >** */tmp/sat.jeff*
2. **sat_interpret** */tmp/sat.jeff* **| more**

Rarely should any user be subjected to this kind of accounting, and this feature should be used carefully and with consideration of the individuals involved.

Access to Particularly Interesting Files or Resources

Sometimes a particular file or resource is of special interest. An information leak may have occurred and an investigation is proceeding into how the leak took place. Or a special file or resource may have been created as bait to trap browsing intruders. In either case, the file or resource should be closely accounted by the auditor.

sat_reduce –n *interesting_file* **–e sat_open –e sat_open_ro** *sat_filename* **|**
sat_interpret

Proper and Improper Management

Frequently, actions taken by the Administrator or root result in unusual audit records. With the enhanced privilege of these accounts, it is not unusual for more audit records of potential concern to be generated. Again, it is rare for a record to be generated that cannot be explained by the normal usage of the system or by simple human error.

Modifications of System Data Files

Every modification of system data files is of interest to the auditor. Since these data files are not only under system security but in fact define system security, any unauthorized access can result in a total breach of security.

Each site has individual policies on how users are added to or removed from the system, how access control of files and hardware is administered, how network connectivity is maintained and administered, and a host of other issues. It is the responsibility of the auditor at each site to enforce the policies of the site and to use the auditing tool effectively to exercise that responsibility.

If you are running Trusted IRIX/B, system data files should be modified only with the dedicated editing tool, *dbedit*, and never with general-purpose text editors. Only privileged users can use the *dbedit* tool, and only privileged users have permission to alter the contents of the system data files. Any use of any other editor on a system data file is a violation of security policy and should be noticed by the auditor. If your interpreted audit trail contains *sat_open* records where the Actual name field contains the string "/secadm," check that the Process ID field (which gives both the PID and the name of the program being executed) does not contain "vi," "ex," "emacs" or any other commonly available text editor. This field should contain only the name "dbedit."

Modifications of Attributes of System Programs

The Administrator should never modify permissions, ownership, or labels of system programs. If your audit trail contains evidence that the administrator has attempted to change attributes of system programs, you should investigate and find the reason for the change. Again, the explanation given is likely to be valid, and this is not good cause to suspect your Administrator of subterfuge; however, you may want to examine your system's security policies and make certain that neither the users nor the administrators take a cavalier attitude toward the security policies.

The following command searches your audit trail for the type of records that can indicate this problem:

```
sat_reduce -e sat_file_attr_write -e sat_fd_attr_write < satfile
```

In the interpreted output, look for lines with the Actual name field. Any audit record showing modified attributes for resources in */bin*, */sbin*, */etc*, */lib*, */var*, */usr/bin*, */usr/lib*, */usr/share*, */usr/bsd*, */usr/sbin*, or */usr/bin/X11* is an audit record deserving follow-up.

Manipulation of the Audit Trail

The auditor should be the only person to access the audit trail. No other users should read from it, write to it, remove files, or modify file attributes. Look at all records generated by people other than the one who knows the auditor account password, and check that none of those records refer to files in */var/adm/sat* or in any other directory you use to store audit trail information.

Archiving Audit Data

Since the audit trail is stored in ordinary system files, archiving your audit data is as easy as making a backup tape. Archive your audit data to conserve disk space but do keep copies of your audit trail; evidence of intrusion and damage to your system may not always be apparent immediately, and the ability to research your audit trail over time can be very valuable in tracking down a security breach. You can use the *compress* utility to reduce the size of your old audit files by up to 80 percent.

Removing Audit Data

Since the audit trail is stored in ordinary system files, once it has been archived, audit trail files can be safely removed. If you enter the *df* command (disk free) and determine that the filesystem containing your audit trail is more than 90 percent full, you should remove old audit files. If your audit files are kept in */var/adm/sat*, enter the command

```
df -k /var/adm/sat
```

The output should be similar to this:

```
Filesystem  Type blocks  use   avail %use Mounted on
/dev/root   efs  245916 218694 27222  89%    /
```

In this example, the file system is 89 percent full, and the auditor should archive and remove audit trail files.

Recovering From Audit File Overflow

Do not allow your audit files to grow too large. Oversized audit files can use up your available disk space and cause the system to refuse new records and immediately cease operations. This can result in lost work and lost audit records. Maintain at least 10 percent free space in your audit filesystem at all times.

The audit daemon, satd(1M), must always be running on your system. The daemon eventually becomes unable to write to the audit file if free disk space drops to 0 percent. When it can no longer write to the audit file, the daemon exits with an error, and the system changes the run level to single-user mode. You must then archive and remove the audit files to free disk space before bringing the system back to multi-user mode. If the *satd* daemon is somehow killed or interrupted on your system, the system changes the run level to single user mode immediately. The daemon is respawned when the system is brought back up.

To make space on the disk for your audit trail, first boot the system into single-user mode. No audit records are generated in this mode. Once in single-user mode, archive your audit files and remove them from the disk. Once at least 10 percent of the filesystem is free, you may boot into multiuser mode without difficulty.

If your auditing system directs the audit files to the / (root) filesystem or the */usr* file system and either filesystem becomes full, you will not be able to bring the system to single-user mode to archive and remove your old audit files. If you find yourself in this situation, perform the following procedures to remove old audit files:

1. Boot the system from the original distribution media, and allow the *inst* utility to start up.
2. At the Inst main menu, select the Admin menu, and then select the *shell* option from the Admin menu. You see a shell prompt.

   From the shell, you must archive and remove the old audit files. Remember that when your system is running the Inst (also called *miniroot*) shell, your system's root directory appears as

   ```
   /root/
   ```

   rather than

```
/
```

and your */usr* file system appears as

```
/root/usr
```

because your system's filesystems are mounted on the Inst filesystem.

3. Once you have created free disk space on your / (root) and */usr* filesystems, you should be able to boot your system normally. If this is a recurring problem, you should refer to the satd(1M) reference page for information on changing the location of your audit files.