SUN BLUEPRINTS [illegible] APPROVED

Auditing in the Solaris™ 8 Operating Environment

By William Osser - Solaris Security Technology Group and Alex Noordergraaf - Enterprise Engineering

Sun BluePrints™ OnLine - February 2001

Sun microsystems
ENTERPRISE ENGINEERING

http://www.sun.com/blueprints

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 806-7759-10
Revision 02, February 2001

Auditing in the Solaris™ 8 Operating Environment

The Solaris™ Operating Environment (Solaris OE) provides the capability to log the activity on a system at a granular level. This logging or auditing ability is part of the Solaris SunSHIELD™ Basic Security Module (BSM). These auditing capabilities were added to provide the features required by the Trusted Computer System Evaluation Criteria (TCSEC) to a security level referred to as C2.

The TCSEC has been superseded by the newer and more internationally recognized Common Criteria security requirements. The Solaris OE has been evaluated under the Controlled Access Protection Profile (CAPP) at Evaluation Assurance Level (EAL) 4. The CAPP used for the Solaris OE evaluation includes all functionality covered by C2 evaluations. For more information about TCSEC and CAPP, refer to the references in the Bibliography.

This article describes the BSM configuration developed for a specific customer. In particular, this article discusses what modifications were made to the default Solaris 8 OE auditing configuration for this customer. Additionally, the process of enabling auditing is discussed.

This article was derived from an auditing case study and includes a set of audit events and classes usable on Solaris 8 OE. The audit events and classes presented in this article are modified from the Solaris 8 OE default configurations. These modified files have been carefully designed to maximize the quality of log data generated while minimizing the quantity. The list of audit events described in this article was generated empirically for a customer, and may not audit all actions appropriate for every situation.

Auditing Principles

One of the main principles of security is accountability; that is, the ability to trace actions taken by a user that could have security relevance. There are some problems associated with accountability, such as the difficulty in determining the security relevance of each user action. Another problem is searching through the collected data to find meaningful information.

An administrator who records selected events in the audit trail and monitors the audit logs is more likely to discover suspicious activity, and thereby prevent a system from being compromised. Additionally, an administrator can review the audit logs to determine what events transpired during an attack. Short of auditing every event exhaustively, there is no way to ensure that all attacks will be discovered. Auditing for Solaris OE merely provides a security conscious administrator one more tool to counter malicious intent.

To audit effectively, an administrator must check the audit trail regularly. If a weakness is exploited in the operating system that is not covered by the current selection of audit events, then the list of audit events being captured must be updated to include such events.

Auditing Goals

The primary goal of auditing is recording user actions to detect malicious intent. Auditing may also act as a deterrent—for example, a malicious user may not take certain actions knowing that those actions will be recorded. The problem is deciding which events generate meaningful information, and which events are so commonplace that they clutter the audit trail. Also, while you may select events to be audited that keep track of all currently known intrusion methods, there may be other methods in which a system can be attacked that are not covered by the selected audit events. Therefore, you should choose audit events that are broad enough to detect potential mischief, but minimal enough so that an administrator is not overwhelmed merely trying to interpret the audit trail.

The secondary goal of auditing is to avoid performance degradation. The more audit events recorded, the greater the load on the system. By minimizing extraneous audit events, the impact on the CPU and I/O subsystems will be reduced. Performance measurements will vary based on system architecture and intended use of the machine.

Many computer installations have attempted to implement auditing, however, it is generally enabled as an afterthought. An administrator examines the list of audit events and picks those that appear relevant, but frequently doesn't test whether this set of events produces comprehensive and useful information. Alternatively, when an integrated solution is delivered to a customer, the last thing to be enabled is auditing, and that is usually performed as the integration team is leaving the building. Implementing auditing in this manner can seriously affect the potential usefulness because the configuration has not been tested, the customer has not been trained on how to manage the configuration, nor has any knowledge been transferred to the customer on how and why the configuration was developed.

Enabling Auditing

Auditing is not enabled by default in standard Solaris OE installations from CD or JumpStart™. If the "JASS" Security Toolkit (described at the end of this article) is used to install the system, there is an option to automatically enable auditing during system installation and configuration. Refer to the Bibliography for additional information.

Two steps are required to enable the BSM functionality on a Solaris OE system. First, the system should be brought to run level 1 (System Maintenance Mode) using the following command:

```
# /usr/sbin/init 1
INIT: New run level: 1
Changing to state 1.
Unmounting remote filesystems: done.
System services are now being stopped.
Dec  6 16:02:09 arrow syslogd: going down on signal 15
Setting netmask of le0 to 255.255.255.0
Killing user processes: done.
Change to state 1 has been completed.

Entering System Maintenance Mode
```

Once in System Maintenance Mode, BSM can be enabled with the following command:

```
# /etc/security/bsmconv
This script is used to enable the Basic Security Module (BSM).
Shall we continue with the conversion now? [y/n] y
bsmconv: INFO: checking startup file.
bsmconv: INFO: move aside /etc/rc2.d/S92volmgt.
bsmconv: INFO: turning on audit module.
bsmconv: INFO: initializing device allocation files.

The Basic Security Module is ready.
If there were any errors, please fix them now.
Configure BSM by editing files located in /etc/security.
Reboot this system now to come up with BSM enabled.
```

When the system is rebooted, a message similar to the following will be displayed during the startup process to indicate that auditing has been enabled:

```
starting audit daemon
Configured 233 kernel events.
```

At this point, auditing is enabled—a log file should be present in the `/var/audit` directory similar to the following:

```
# ls -l /var/audit
-rw------- 1 root root 27947 Dec 6 16:15 20001206211216.not_terminated.arrow
```

If BSM functionality is no longer required on a Solaris OE system, it can be disabled using the `bsmunconv` command.

Note – When the `bsmconv` command is run, it disables the Stop-a keyboard sequence by adding `set abort_enable = 0` to the `/etc/system` file. Disabling the ability of a user or administrator to stop a system through a keyboard Stop-a or equivalent command over a serial port may not be appropriate for all environments.

Definition of Terms

Before discussing BSM configuration details, several terms need to be defined. This section defines the auditing terms and discusses their default values in the Solaris 8 OE BSM. These terms are essential for understanding and successfully deploying the auditing process.

Audit Flag

The official BSM manual, *"SunSHIELD Basic Security Module Guide,"* available from `http://docs.sun.com` uses *audit flags* only to define event selection, and uses *audit class* only to define the audit classes available on the system.

Audit Preselection Mask

The *Audit Preselection Mask* is a 32 bit field representing the logical sum of all audit classes being audited by a process. When a user first connects to the system through `login`, the `audit_control` file is reviewed to determine the audit classes to be enabled. Any additional audit classes set in the `audit_user` file for that user (using the Audit UID) are also added to the audit mask. Unless a process is explicitly assigned a new audit mask, the process inherits the audit mask of its parent.

Audit Trail

The audit trail is the set of audit log files that have been recorded by the system. The audit trail can be analyzed with the use of the `auditreduce` and `praudit` commands. The `dir:` parameter in the `audit_control` file specifies where these logs are stored.

Audit User ID (AUID)

When BSM is enabled, a third ID, the Audit User ID (or Audit ID), becomes enabled. The Audit ID is set during login authentication and does not change for the duration of the session. Actions such as `su`, which change the real or effective UID, do not change the Audit ID. The Audit ID is recorded in the audit trail with each audit event being recorded, thereby tying actions to the user who was authenticated at login regardless of which `su` or `setuid` actions occur.

`audit_class`

An audit class is a group of audit events. All audit classes are defined in the `/etc/security/audit_class` file. All audit events are assigned to audit classes in the`/etc/security/audit_event` file. Audit classes are recorded in the audit trail if they are turned on globally in the `audit_control` file, or are assigned to a specific user in the `audit_user` database.

These audit classes are used by the `audit_control`, `audit_user`, and `audit_event` files, as well as in the audit mask.

All entries in the `audit_class` file have the following format:

mask:name:description

Where mask, or class mask, uses an unsigned integer to represent the class mask; providing a total of 32 different classes. The name field, or audit class name, is a two character mnemonic of the class name used to represent that class in the other audit configuration files. The description field provides a mechanism to document what the class definition represents.

As delivered in Solaris 8 OE, all audit classes are one bit, but a hierarchy can be set up by using more than one bit. The following example illustrates `audit_class` definitions using only one bit for the audit mask:

```
0x00000010:fc:file create
0x00000020:fd:file delete
0x00000040:cl:file close
```

These three classes, which are defined in the `audit_class` file shipped with Solaris 8 OE, can be combined into one super audit class as illustrated below:

```
0x00000070:ff:file access
```

The mnemonic `fa` is already used for `file attribute access`, therefore, `ff` was chosen instead. Note that this audit class definition would generate large volumes of data on most systems.

audit_control

The `audit_control` file describes system parameters for auditing. Among these parameters are, the audit trail storage directory (or directories), the minimum free space warning value, and the audit flags assigned to user and system processes.

It is possible to audit only failed, or only successful audit events as desired. For example, a successful attempt to allocate memory should not be recorded, but a failed attempt should be recorded. This can be specified in either the `audit_control` or `audit_user` files.

The default Solaris 8 OE `audit_control` file contains the following:

```
dir:/var/audit
minfree:20
flags:
naflags:
```

For additional information on these parameters, refer to the `audit_control` man page.

audit_event

Audit events are assigned to specific actions in the operating system. Some are low level, having to do with calls to the kernel; and some are high level, having to do with programs such as login. Audit events may be assigned to more than one audit class.

There are two categories of audit events: *kernel* level and *user* level. Kernel level events are those events ranging from 1 to 2047. User level audit events are numbered from 2048 to 65535. Events defined by a third party should be assigned to the upper range of audit events, 32768 to 65535.

Note – The action of reviewing an audit log can also be an auditable event.

If an audit event is in an audit class that is in the process audit mask, then an audit record is generated and added to the audit trail.

The audit_event file, located in /etc/security, defines the audit_events and assigns each event to one or more audit classes. Software developers can define new audit events. Upon doing so, they must also be defined in either /usr/include/bsm/audit_uevents.h or /usr/include/bsm/audit_kevents.h. As a general rule, a software developer should not add a kernel event.

For additional information on the audit_event file, refer to the audit_event man page.

audit_user

The audit_user database provides the capability to specify additional auditing for individual users. Access to this database follows the rules for the password database specified in /etc/nsswitch.conf. The audit_user file is located in /etc/security. The *Audit Preselection Mask* is generated by combining a user's audit_user entry, if any, and the machine specific audit flags contained in /etc/security/audit_control.

Audit Trails

One question frequently asked is, "How much space is required for the audit trail"? The answer to this question is difficult to quantify because there are several factors which have significant impact on the quantity of audit logs, including but not limited to:

- The frequency of audit log examination and archiving
- The amount of auditing options enabled
- The system activity
- The amount of traffic generated between each examination and archiving

In 1993, a Sun™ customer proudly presented their 26 Gbyte disk farm that was dedicated solely to the audit trail. That was certainly a generous amount in 1993, however, computers today are capable of accomplishing far more in a shorter time span. The amount of space necessary for auditing depends upon the environment in which auditing is to be used. When the configuration recommended by this article was first deployed, analysis showed that less then 2 Gbyte of audit data storage was required. A total of 4 Gbyte was made available for the storage of audit trail information to make provisions for unexpected events.

To determine how much space is necessary, start by making educated choices about audit events to record, and then examine test results to determine if the correct amount of data is being generated. The process of auditing requires frequent checking of the audit logs to determine if any assault has occurred, and to archive and remove old audit trails to avoid overloading the file system.

One method for determining how much space is required for the audit trail is based on the following assumptions:

- The audit trail is examined twice a day to determine if any damage, malicious or inadvertent, has been done to the system, and to determine how much audit data has been recorded.
- The audit trail is archived once each week. Once archived, the old audit trail is removed.

If the average amount of audit data per 12 hours is 5 Mbyte, the archive process would involve 70 Mbyte. A good rule of thumb is to provide at least twice this amount of space. However, if the standard deviation from the mean is greater than the average, this space may need to be quadrupled to 280 MB. Given the size of modern storage sub-systems and the negative impact of the audit logging partition filling up, setting aside additional space is recommended.

Ideally, multiple audit log storage partitions should be available to any system performing auditing. If using NFS, then a local audit log partition should be available in case of network outage. Additionally, the `minfree` option in the `audit_control` file should be used. If a disk partition used for audit log storage has less then `minfree` percent disk space available, the `auditd` will generate an e-mail to the address specified by `audit_warn`.

The default configuration of BSM is to not record audit events when all available audit log storage space is used. However, a count of the number of audit events dropped will be kept. In this configuration, the system will continue functioning normally while not generating audit events until more space is made available to record the audit data.

This default configuration is defined by the `bsmconv` script that is run to enable BSM on a system. When run, the `bsmconv` script creates the `/etc/security/audit_startup` script. This script, contains the following:

```
#!/bin/sh
auditconfig -conf
auditconfig -setpolicy none
auditconfig -setpolicy +cnt
```

By specifying `setpolicy +cnt`, the `audit_startup` script created by `bsmconv` forces the auditing subsystem to drop auditable events while keeping a count of the total number of events dropped.

If the desired configuration is to suspend all processes when the audit partitions are full, the line:

```
auditconfig -setpolicy +cnt
```

should be modified to:

```
auditconfig -setpolicy -cnt
```

The `+cnt` must be changed to `-cnt` and the system rebooted.

It is important to archive and remove old audit trail data to avoid a full audit trail. This avoids a possible denial-of-service (DoS) attack, which would attempt to deliberately fill all available audit log storage space. Similarly, auditing of network events should be scrutinized routinely because an intruder could generate an enormous amount of audit data in a DoS attack.

Note – One of the issues with audit log files is that when a root compromise occurs, the audit trail should no longer be trusted. Audit logs are stored in a binary format, but this format provides no protection against modification by unauthorized individuals.

Audit Classes and Events

This section describes a selection of audit events used in an auditing case study. With the evolution of operating system security, as well as system attacks, this list will change periodically. Also, some of the audit events listed in this section may not be necessary at every site, so they may be excluded to reduce the amount of traffic in the audit logs.

The audit events described in this section are grouped by audit class. For each audit class, the goal of the class is described in addition to any audit events that were added or deleted from the default list.

Any audit event removed from an audit class is assigned to an unselected audit class (e.g., `no`). In this way, no audit record is generated when that event occurs.

Note – This Sun BluePrints™ OnLine article is specific to Solaris OE version 8. Differences that may be encountered between Solaris 8 OE updates are mentioned. Changes between previous Solaris OE versions and Solaris OE version 8 are not discussed.

Login or Logout (lo)

This audit class captures all attempts (failed or successful) to login to the system. The following is the list of audit events that map to this audit class:

- 6152:AUE_login:login - local:lo
- 6153:AUE_logout:logout:lo
- 6154:AUE_telnet:login - telnet:lo
- 6155:AUE_rlogin:login - rlogin:lo
- 6158:AUE_rshd:rsh access:lo
- 6159:AUE_su:su:lo
- 6162:AUE_rexecd:rexecd:lo
- 6163:AUE_passwd:passwd:lo
- 6164:AUE_rexd:rexd:lo
- 6165:AUE_ftpd:ftp access:lo
- 6213:AUE_admin_authenticate:adminsuite access:lo

This audit class should almost always be selected by any site that implements auditing because the audit class records both the start and end of user interactions with the system.

Note – Event 6213 is a relatively new audit event, having been added in Solaris 8 OE 10/00. If you have an earlier version of the Solaris 8 OE that has not been updated (with the latest patches), you may not find this audit event.

Non-attribute (na)

This audit class captures non-attributable audit events. Notice the audit events that map to this class:

- 113:AUE_SYSTEMBOOT:system booted:na
- 153:AUE_ENTERPROM:enter prom:na
- 154:AUE_EXITPROM:exit prom:na
- 6151:AUE_inetd_connect:inetd:na
- 6156:AUE_mountd_mount:mount:na
- 6157:AUE_mountd_umount:unmount:na

All sites should have an audit record of when a system was booted, and also whenever anyone enters the PROM mode. Mounting and unmounting of file systems should also be recorded. All sites should also record connections to the `inetd`. The `inetd` provides access to system services, and should be monitored. If this is the primary service that your system is providing, this is the precise area that should be audited. If this audit event generates more information than can be adequately reviewed and analyzed during the time allotted for audit review, it is recommended that the amount of time dedicated to audit review is increased.

None of these events should generate an excessive amount of auditing information—unless your system is frequently rebooted, with the possible exception of the `inetd_connect` event.

Administrative (ad)

This audit class is for administrative actions, which can cover a variety of actions. The following is the audit events that map to this audit class:

- 9:AUE_MKNOD:mknod(2):ad
- 12:AUE_UMOUNT:umount(2) - old version:ad
- 18:AUE_ACCT:acct(2):ad
- 20:AUE_REBOOT:reboot(2):ad
- 28:AUE_SWAPON:swapon(2):ad
- 29:AUE_SETHOSTNAME:sethostname(2):ad
- 37:AUE_SETTIMEOFDAY:settimeofday(2):ad
- 50:AUE_ADJTIME:adjtime(2):ad
- 53:AUE_NFS_SVC:nfs_svc(2):ad
- 56:AUE_UNMOUNT:unmount(2):ad
- 57:AUE_ASYNC_DAEMON:async_daemon(2):ad
- 59:AUE_SETDOMAINNAME:setdomainname(2):ad
- 60:AUE_QUOTACTL:quotactl(2):ad
- 61:AUE_EXPORTFS:exportfs(2):ad
- 62:AUE_MOUNT:mount(2):ad
- 114:AUE_ASYNC_DAEMON_EXIT:async_daemon(2) exited:ad
- 115:AUE_NFSSVC_EXIT:nfssvc(2) exited:ad
- 131:AUE_SETAUID:setauid(2):ad
- 133:AUE_SETAUDIT:setaudit(2):ad
- 135:AUE_SETUSERAUDIT:setuseraudit(2):ad
- 136:AUE_AUDITSVC:auditsvc(2):ad
- 140:AUE_AUDITON_STERMID:auditon(2) - SETTERMID command:ad
- 142:AUE_AUDITON_SPOLICY:auditon(2) - SPOLICY command:ad
- 144:AUE_AUDITON_SESTATE:auditon(2) - SESTATE command:ad
- 146:AUE_AUDITON_SQCTRL:auditon(2) - SQCTRL command:ad
- 148:AUE_SETKERNSTATE:setkernstate(2):ad
- 150:AUE_AUDITSTAT:auditstat(2):ad
- 201:AUE_STIME:old stime(2):ad

- 222:AUE_AUDITON_SETKMASK:auditon(2) - set kernel mask:ad
- 226:AUE_AUDITON_SETSTAT:auditon(2) - reset audit statistics:ad
- 227:AUE_AUDITON_SETUMASK:auditon(2) - set mask per uid:ad
- 228:AUE_AUDITON_SETSMASK:auditon(2) - set mask per session ID:ad
- 230:AUE_AUDITON_SETCOND:auditon(2) - set audit state:ad
- 232:AUE_AUDITON_SETCLASS:auditon(2) - set event class:ad
- 233:AUE_UTSSYS:utssys(2) - fusers:ad
- 243:AUE_MODLOAD:modctl(2) - load module:ad
- 244:AUE_MODUNLOAD:modctl(2) - unload module:ad
- 245:AUE_MODCONFIG:modctl(2) - configure module:ad
- 246:AUE_MODADDMAJ:modctl(2) - bind module:ad
- 262:AUE_P_ONLINE:p_online(2):ad
- 263:AUE_PROCESSOR_BIND:processor_bind(2):ad
- 266:AUE_SETAUDIT_ADDR:setaudit_addr(2):ad
- 268:AUE_UMOUNT2:umount2(2):ad
- 6144:AUE_at_create:at-create atjob:ad
- 6145:AUE_at_delete:at-delete atjob (at or atrm):ad
- 6146:AUE_at_perm:at-permission:ad
- 6147:AUE_cron_invoke:cron-invoke:ad
- 6148:AUE_crontab_create:crontab-crontab created:ad
- 6149:AUE_crontab_delete:crontab-crontab deleted:ad
- 6150:AUE_crontab_perm:crontab-permisson:ad
- 6160:AUE_halt_solaris:halt(1m):ad
- 6161:AUE_reboot_solaris:reboot(1m):ad
- 6166:AUE_init_solaris:init(1m):ad
- 6167:AUE_uadmin_solaris:uadmin(1m):ad
- 6168:AUE_shutdown_solaris:shutdown(1b):ad
- 6169:AUE_poweroff_solaris:poweroff(1m):ad
- 6170:AUE_crontab_mod:crontab-modify:ad
- 6200:AUE_allocate_succ:allocate-device success:ad
- 6201:AUE_allocate_fail:allocate-device failure:ad
- 6202:AUE_deallocate_succ:deallocate-device success:ad
- 6203:AUE_deallocate_fail:deallocate-device failure:ad
- 6205:AUE_listdevice_succ:allocate-list devices success:ad
- 6206:AUE_listdevice_fail:allocate-list devices failure:ad
- 6207:AUE_create_user:create user:ad
- 6208:AUE_modify_user:modify user:ad
- 6209:AUE_delete_user:delete user:ad
- 6210:AUE_disable_user:disable user:ad
- 6211:AUE_enable_user:enable user:ad

Note – Events 265 through 268 are relatively new audit events. Events 265 through 267 where added in first release of Solaris 8 OE, while event 268 was added in Solaris 8 OE 6/00. If you have an earlier version of the Solaris 8 OE that has not been updated (with the latest patches), you may not find these audit events.

Note – Solaris 8 OE defines `AUE_MKNOD` to be an event in audit class `fc`. The recommendation of this article is that this event is more appropriately an Administrative, or `ad` audit class.

Almost all of these audit events happen infrequently—for example, only on system shutdown or startup. Some may occur more frequently, such as modifications to the `cron` tables or `at` jobs. Each site must determine whether these modifications should be recorded.

There are six audit events that handle device allocation, which is another feature included in the BSM. If your site is not using device allocation, then including these audit events generates no additional information. If your site is using this feature, these audit events will keep track of which users have used these devices. If it is discovered that data was removed from the system using peripheral devices, these audit events will describe which users used those devices.

The Solaris 8 OE is delivered with many other audit events mapped to the `ad` class. The events listed below generated excessive information and were re-mapped to another class that was not being used:

- 51:AUE_SETRLIMIT:setrlimit(2):no
- 138:AUE_AUDITON:auditon(2):no
- 239:AUE_SYSINFO:sysinfo(2):no

The following event should be excluded because it can lead to excessive audit data. However, if your site's usage of NFS is limited, then this event may be added:

- 58:AUE_NFS_GETFH:nfs_getfh(2):no

While obtaining audit information is considered a privileged action, information from the following audit events does not provide anything adequately meaningful. In addition, some of these events are generated frequently. The following events should be excluded at most sites:

- 130:AUE_GETAUID:getauid(2):no
- 132:AUE_GETAUDIT:getaudit(2):no
- 134:AUE_GETUSERAUDIT:getuseraudit(2):no
- 139:AUE_AUDITON_GTERMID:auditon(2) - GETTERMID command:no
- 141:AUE_AUDITON_GPOLICY:auditon(2) - GPOLICY command:no
- 143:AUE_AUDITON_GESTATE:auditon(2) - GESTATE command:no
- 145:AUE_AUDITON_GQCTRL:auditon(2) - GQCTRL command:no
- 147:AUE_GETKERNSTATE:getkernstate(2):no
- 149:AUE_GETPORTAUDIT:getportaudit(2):no
- 221:AUE_AUDITON_GETKMASK:auditon(2) - get kernel mask:no
- 223:AUE_AUDITON_GETCWD:auditon(2) - get cwd:no
- 224:AUE_AUDITON_GETCAR:auditon(2) - get car:no
- 225:AUE_AUDITON_GETSTAT:auditon(2) - get audit statistics:no
- 229:AUE_AUDITON_GETCOND:auditon(2) - get audit state:no

- 231:AUE_AUDITON_GETCLASS:auditon(2) - get event class:no
- 267:AUE_GETAUDIT_ADDR:getaudit_addr(2):no

A more security conscious site may wish to audit these events when the event itself fails to execute properly. A `getaudit` event should never fail in a normal system. If a `getaudit` event does fail, it could indicate probing and should be tracked. To audit `getaudit` event activity, all the audit events listed above should be assigned to a new audit class and audited only upon failure. For example, a new audit class `ag` would be added to the `audit_class` file by an administrator. An additional audit flag, `-ag`, must also be added to the `flags:` line in the `audit_control` file.

Additional Audit Events

To avoid too much modification of the `audit_event` file, two new audit classes were created from audit classes not used by the default auditing configuration. These new audit classes are a combination of the `fm` file attribute modify and `pc` process classes, and are described in greater detail below.

Custom Audit Events (cs)

This audit class captures file ownership changes, change root directories (`chroot`), process priority settings, and changes to a process UID. This group catches security relevant events that are not captured by other audit classes. The events in this audit class are listed below:

- 11:AUE_CHOWN:chown(2):fm,cs
- 24:AUE_CHROOT:chroot(2):pc,cs
- 31:AUE_SETPRIORITY:setpriority(2):pc,cs
- 38:AUE_FCHOWN:fchown(2):fm,cs
- 40:AUE_SETREUID:setreuid(2):pc,cs
- 69:AUE_FCHROOT:fchroot(2):pc,cs
- 200:AUE_OSETUID:old setuid(2):pc,cs
- 203:AUE_NICE:old nice(2):pc,cs
- 212:AUE_PRIOCNTLSYS:priocntlsys(2):pc,cs
- 215:AUE_SETEUID:seteuid(2):pc,cs
- 237:AUE_LCHOWN:lchown(2):fm,cs

Note – The original mapping of the audit event to class has been retained. This will ease migration to a newer version of Solaris OE. This produces no additional audit data because neither the `fm` nor `pc` audit classes are being audited.

Custom Ancillary Audit Events (cf)

These are additional events, to be captured, as part of the recommendations made in this article. It may not be necessary to capture these events in a site that has fewer security requirements; if unnecessary this class can be disabled:

- 10:AUE_CHMOD:chmod(2):fm,cf
- 39:AUE_FCHMOD:fchmod(2):fm,cf
- 251:AUE_ACLSET:acl(2) - SETACL command:fm,cf
- 252:AUE_FACLSET:facl(2) - SETACL command:fm,cf

These events note changes to file attributes that may or may not require privilege. Frequently an *intruder* would need to change file attributes in order to abuse the system while *users* do not normally change file permissions. Note that a recursive `chmod` can generate a large amount of audit information.

Application Audit Class

An audit class is reserved for applications that generate audit data and should be enabled by default in case the application generates audit data. It may be discovered later that certain application audit events generate too much information. In such cases, those audit events should be re-mapped to the `no` class. If there are no applications that interact with the audit system, this audit class will generate no additional audit data.

Excluded Audit Classes

Several audit classes were excluded because they did not generate meaningful information (or generated too much data).

The following audit classes were not included:

- 0x00000001:fr:file read
- 0x00000002:fw:file write
- 0x00000004:fa:file attribute access
- 0x00000008:fm:file attribute modify
- 0x00000010:fc:file create
- 0x00000020:fd:file delete
- 0x00000040:cl:file close
- 0x00000080:pc:process
- 0x00000100:nt:network
- 0x00000200:ip:ipc
- 0x20000000:io:ioctl
- 0x40000000:ex:exec
- 0x80000000:ot:other

Each of these excluded audit classes generated significant amounts of data which was not particularly useful. The configuration recommended in this article already audits a number of events from the `process` audit class. The `network` and `ipc` audit classes may be useful in some environments, but will not be included in this auditing configuration. The `ioctl` class generates a large volume of data. The `exec` audit class will record all processes that started, which may be useful. Each site must determine if these audit classes generate useful information.

Audit Trail Analysis

Audit analysis was one of the problems encountered because the audit review process can generate significant quantities of audit log data. Whenever the audit trail is reviewed, the audit trail would grow to include more noise than information (from the perspective of the audit trail reviewer). One possible solution is to create a special account that is not audited for events in the `ad` class; that is, the entry for this user in the `audit_user` file excludes the `ad` audit flag. The user performing audit review would log in through this special account, `su` to become the super-user, and review the audit trail thereby minimizing the creation of audit data.

The introduction of Role Based Access Control (RBAC) in the Solaris 8 OE permits an audit administration role to be created that would only allow super-user access for audit review tasks. The steps involved in implementing this recommendation are not included in this article.

`audit_control`, `audit_class`, and `audit_event` Files

This section briefly describes the modifications made to the `audit_control`, `audit_class`, and `audit_event` files discussed in this article. These files implement all the recommendations made in this article. These files are available from:

`http://www.sun.com/blueprints/tools`.

audit_control File

The naflags, or non-attributable flags field is currently used by events not mapped to a particular user. These events include login programs (for example, dtlogin), and other programs launched during system boot (for example, mountd and inetd). Only the lo and ad flags should be in the naflags field. Programs that have not set their audit context do not generate audit events. The complete audit_control file is as follows:

```
dir:/var/audit
flags:lo,na,ad,cs,cf,ap
minfree:20
naflags:lo,ad
```

Modified audit_class File

The only difference between the file presented here and the one provided with Solaris 8 OE is the addition of two audit classes: cf and cs. For example:

```
0x01000000:cs:custom audit events
0x02000000:cf:custom ancillary audit events
```

Modified audit_event File

This file has been modified extensively. For the original version, please reference the file /etc/security/audit_event on a machine running Solaris 8 OE.

audit_event Modifications

Note – When modifying the audit_event file, care must be taken to ensure that no blank lines are present. If a blank line is present, the praudit command will be unable to parse terminated audit log files.

Specifically, an error similar to the following may be produced when using praudit to review audit log files:

```
# praudit 20010109175138.20010109190123.williams
file,Tue Jan 09 12:51:38 2001, + 52084 msec,
header,85,2,Segmentation Fault
```

Solaris OE Upgrades

This article made recommendations to re-map some audit events to other audit classes. When the next release of Solaris OE is installed, the audit event mappings may change. It is also possible that the audit classes may change. When upgrading to a newer release of Solaris OE, the existing audit_event and audit_class files must be merged. Any audit events added in the new release must be examined to determine if they should be included in the set of auditable events.

JumpStart Architecture and Security Scripts “JASS” Toolkit

The recommendations made in this article have been included in the “JASS” Security Toolkit version 0.3. This version of the “JASS” Toolkit includes scripts to enable auditing and modify the default Solaris OE auditing configuration to the configuration described in this article. The “JASS” Security Toolkit may be downloaded from:

`http://www.sun.com/blueprints/tools`

Summary

No system is foolproof. Enabling auditing will not prevent intrusions, nor will auditing necessarily provide details on the attacks used. An attacker could turn off all audit flags in the audit mask, or turn off auditing entirely while removing the audit trail. Auditing enables an administrator to detect probes and attacks, and assists in taking appropriate action and prevent future attacks.

Good auditing practice is an ongoing process, not a state. Findings from the audit review process should include not only the detection of possible attacks, but also information that enables an administrator to decide which events should and should not be recorded.

Implementing the recommendations made in this article provides a comprehensive auditing environment for Solaris 8 OE systems. However, this article should only be the starting point in the development of appropriate auditing processes and procedures.

Bibliography

Noordergraaf, Alex and Brunette, Glenn, *Jumpstart Architecture and Security Scripts Toolkit - Part 1 - Updated for version 0.2*, Sun BluePrints OnLine, November 2000.
`http://www.sun.com/blueprints/1100/jssec-updt1.pdf`

Noordergraaf, Alex and Brunette, Glenn, *Jumpstart Architecture and Security Scripts Toolkit - Part 2 - Updated for version 0.2*, Sun BluePrints OnLine, November 2000.
`http://www.sun.com/blueprints/1100/jssec2-updt1.pdf`

Noordergraaf, Alex and Brunette, Glenn, *Jumpstart Architecture and Security Scripts Toolkit - Part 3 - Updated for version 0.2*, Sun BluePrints OnLine, November 2000.
`http://www.sun.com/blueprints/1100/jssec3-updt1.pdf`

SunSHIELD Basic Security Module Guide, Sun Microsystems,
`http://docs.sun.com`

SunSolve InfoDoc 17361: *Using BSM to Log All Commands Run By A User*,
http://sunsolve.sun.com

TCSEC and Interpretations,
http://www.radium.ncsc.mil/tpep/library/tcsec/index.html

Trusted Technology Assessment Program's explanation of Common Criteria
http://www.radium.ncsc.mil/tpep/library/ccitse/index.html

Other References

Moffat, Darren J., *Solaris BSM Auditing*,
http://www.securityfocus.com/focus/sun/articles/bsmaudit1.html

Porras, Phillip A., Neumann, Peter G., *EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances*, Computer Science Laboratory SRI International,
http://www.csl.sri.com/intrusion.html

University of California, Davis, Computer Science Department, *The Audit Workbench Project*, http://seclab.cs.ucdavis.edu/awb/AuditWorkBench.html

Author's Bio: William Osser

Will Osser has over 8 years experience in the area of Computer and Network Security. He has worked extensively with B-1 secure Unix systems in a variety of roles including developing, sustaining, pre- and post-sales support, as well as training. He has also worked as a security consultant designing system and software architecture. Will is currently a software engineer working for Sun Microsystems in the Solaris Secure Technology Group.

Will joined Sun directly after completing his Master's Thesis in Computer Engineering at the University of California.

Author's Bio: Alexander Noordergraaf

Alex Noordergraaf has over nine years experience in the area of Computer and Network Security. As a Senior Staff Engineer in the Enterprise Engineering group of Sun Microsystems he is developing, documenting, and publishing security best practices through the Sun BluePrints OnLine program. Recently he released the freeware "JASS" Security Toolkit, which is available from http://www.sun.com/blueprints/tools.

Prior to his role in Enterprise Engineering he was a Senior Security Architect with Sun Professional Services where he worked with many Fortune 500 companies on projects that included security assessments, architecture development, architectural reviews, and policy/procedure review and development. In addition to providing billable services to customers he developed and delivered an Enterprise security assessment methodology and training curriculum to be used worldwide by SunPS. His customers have included major telecommunication firms, financial institutions, ISPs, and ASPs.