Auditing System Security

Alex Noordergraaf, Enterprise Server Products

Glenn Brunette, Sun Professional Services

Sun BluePrints™ OnLine—May 2003

http://www.sun.com/blueprints

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95045 U.S.A.
650 960-1300

Part No. 817-2881-10
Revision 1.0, 5/6/03
Edition: May 2003

Please
Recycle

Auditing System Security

Editor's Note – This article is the complete sixth chapter of the Sun BluePrints™ book, *Securing Systems With the Solaris Security Toolkit*, by Alex Noodergraaf and Glenn Brunette (ISBN 0-13-141071-7), which is available through `www.sun.com/books`, amazon.com, and Barnes & Noble bookstores in late June or early July.

This chapter describes how to audit (validate) a system's security using the Solaris Security Toolkit software. Use the information and procedures in this chapter for maintaining an established security profile after hardening. For systems that are already deployed, you may want to use the information in this chapter to assess security before hardening.

Note – The term *audit* is used in this chapter and book to define the Solaris Security Toolkit software's automated process of validating a security posture by comparing it with a predefined security profile. The use of this term in this publication does not represent a guarantee that a system is completely secure after using the audit option.

This chapter contains the following topics:

Maintaining Security

Maintaining security is an ongoing process and is something that must be reviewed and revisited periodically. Maintaining a secure system requires vigilance, because the default security configuration for any system tends to become increasingly open over time. (For more information about maintaining security, refer to Chapter 2, “Maintaining System Security” on page 36.)

Based upon user experience and requests, we developed an automated method for the Solaris Security Toolkit software to audit the security posture of a system, by determining its level of compliance with a specified security profile.

Note – This method is only available in standalone mode using the `jass-execute -a` command and cannot be used during a JumpStart installation.

We recommend that you audit the security posture of your systems periodically, either manually or automatically (for example, via `cron` job or an `rc` script). For example, after hardening a new installation, execute the Solaris Security Toolkit software audit command (`jass-execute -a <driver-name>`) five days later to determine if the system security has changed from the state defined by the security profile.

How often you audit security depends on the criticality of the environment and your security policy. Some users run an audit every hour, every day, or only once a month. Some users run a mini-scan (limited number of checks) every hour, and a full scan (with all the possible checks) once a day.

Consider auditing an essential component to maintain the security posture of deployed systems. If security posture is not periodically audited, then configurations often drift over time due to entropy or modifications that unknowingly or maliciously change the desired security posture. Without periodic review, these changes go undetected and corrective measures are not taken. The result is a system that becomes less secure and, correspondingly, more vulnerable.

In addition to periodic audits, we recommend that you perform audits after upgrades, patches, and other significant system configuration changes.

Reviewing Security Prior to Hardening

In some cases, you may find it useful to review the security posture on deployed systems *before* hardening them. For example, if you assume responsibility for deployed systems that another person administrated, inspect the state of the systems so that you know their posture and, if necessary, can bring them into compliance with the same security profiles used on your other systems.

Another example that commonly applies is when a consultant, such as a Sun Professional Services consultant, wants to determine the security posture of a deployed system for a customer before securing the system. In this scenario, the consultant typically executes one of the Solaris Security Toolkit security profiles in audit mode to determine what changes would be made to a system without actually making the changes. Of course, without customizing the security profile, the result is a high-water mark, and the output might contain false-positive vulnerabilities. However, consultants may find the output useful as a starting point from which to develop and implement custom security profiles for the customer's systems.

Customizing Security Audits

The audit option provides a highly flexible and extensible mechanism for evaluating the state of a system. As with hardening scripts, you can customize the actions of audit scripts. For example, you can customize environment variables, customize framework and helper functions, add new checks, and add functionality to the audit framework.

Typically, most users find the standard and product-specific audit scripts are suitable as templates from which to customize auditing for their environments. For this scenario, customize audit script actions through drivers, finish scripts, environment variables, and file templates. These custom changes can be made with little effort and without modifying the code. Whatever changes you make for hardening are automatically known by the Solaris Security Toolkit software when you perform auditing.

Occasionally, some users find it necessary to add checks or functionality that the Solaris Security Toolkit software does not provide. For this scenario, add the checks or new functionality to the audit script. (You may want to make related changes in the corresponding finish script.) In some cases, you may need to modify the code. Use extreme care when performing code additions and modifications, to avoid introducing bugs and failures.

Rarely, some users find that they need to create entirely new proprietary, or site-specific, drivers and scripts. For this scenario, we recommend that you use the templates and samples as guidelines when coding the new drivers and scripts. Also, be advised that site-specific drivers, finish scripts, variables, and functions are *not* automatically known to the Solaris Security Toolkit software when you use the audit option. For example, if you add a site-specific driver named `abcc-nj-secure.driver` that contains a site-specific finish script, `abcc-nj-install-foo.fin`, then you need to create a site-specific audit script, `abcc-nj-install-foo.aud`. Similarly, if you start with only the audit script, you should create the matching finish script.

To customize or create new drivers, scripts, variables, and functions, use the following information:

- For drivers, refer to Chapter 10.
- For finish scripts, refer to Chapter 11.
- For audit scripts, refer to Chapter 12.
- For variables, refer to Chapter 13.
- For functions, Chapter 8.

For example, what if you need to add a patch that the Solaris Security Toolkit software does not install? You can extend one of the standard or product-specific templates, or you can create your own. If you create your own, create a finish script to add the patch, then create the corresponding audit script to check for the patch installation.

Preparing to Audit Security

To use the instructions and recommendations in this chapter, you need a security profile. For information about developing and implementing a security profile, refer to Chapter 2.

A variety of security profile templates are included with the Solaris Security Toolkit distribution as drivers. As mentioned earlier in this book, the default security profile and changes made by these drivers might not be appropriate for your systems. Typically, the security profiles implemented by these drivers are “high-water” marks for security. By this, we mean that they disable services that are not required, and they enable optional security features disabled by default.

Many Solaris Security Toolkit software users find that the standard and product-specific security profile templates are acceptable for their environments. If this applies to your situation, then determine which security profile is closest to the security posture you want, and use it for both assessing and hardening your systems.

The preferred practice we recommend, however, is that you review and customize the security profile templates for your environment, or develop new ones. Techniques and recommendations for customizing security profiles are provided in Chapter 10. This approach provides a security posture tailored for your organization, and it minimizes the amount of false errors returned during a security assessment. For example, if you know that Telnet needs to be enabled, you can customize the security profile so that when performing a security assessment, the software does not consider Telnet a vulnerability. For example, a site using Telnet with Kerberos, for authentication and encryption, would not consider the use of Telnet a vulnerability.

Using Options and Controlling Audit Output

This section describes the options available for executing an audit run and the options for controlling output. This section contains the following topics:

Command Line Options

Example usage to audit a system against a security profile:

```
# jass-execute -a driver [ -V verbosity ] [ -q | -o output_file ]
[ -m e-mail_address ]
```

When executing the Solaris Security Toolkit software audit command, you can use the following options listed in TABLE 0-1.

TABLE 0-1 Using Command Line Options With the Audit Command

Option	Description
`-h`	Displays the `jass-execute` help message, which provides an overview of the available options.
`-m`	Mails output to an email address.
`-o`	Directs output into a file.
`-q`	Prevents the display of output to the console. Also known as the quiet option.
`-V`	Specifies the verbosity level for an audit run.

For detailed information about the options available with `jass-execute -a` command, refer to the following sections:

- “Quiet Option” on page 9
- “Verbosity Option” on page 9

Display Help Option

The `-h` option displays the `jass-execute` help message, which provides an overview of the available options.

The `-h` option produces output similar to the following:

CODE EXAMPLE 0-1 Sample `-h` Option Output

```
# ./jass-execute -h

To apply this Toolkit to a system, using the syntax:
   ./jass-execute [-r root_directory -p os_version ]
[ -q | -o output_file ] [ -m e-mail_address ] [-d] driver

To undo a previous application of the Toolkit from a system:
   ./jass-execute -u [ -n ] [ -q | -o output_file ]
[ -m e-mail_address ]

To audit a system against a pre-defined profile:
   ./jass-execute -a driver [ -V verbosity ]
[ -q | -o output_file ] [ -m e-mail_address ]

To display the history of Toolkit applications on a system:
   ./jass-execute -H

To display the last application of the Toolkit on a system:
   ./jass-execute -l

To display this help message:
   ./jass-execute -h
```

Email Notification Option

The `-m <email address>` option provides a mechanism by which standalone hardening and undo output can be emailed automatically by the Solaris Security Toolkit software when the run completes. The email report is in addition to any logs generated on the system using other options.

A Solaris Security Toolkit run calling `sunfire_15k_sc-config.driver` using the email option would be similar to the following:

```
# ./jass-execute -m root -d sunfire_15k_sc-config.driver
[...]
```

Output File Option

The `-o <output_file>` option redirects the console output of `jass-execute` runs to a separate file, `output_file`.

This option has no effect on the logs kept in the `JASS_REPOSITORY` directory. This option is particularly helpful when performed over a slow terminal connection, because there is a significant amount of output generated by a Solaris Security Toolkit run.

This option can be used with either the `-d`, `-u`, or `-a` options.

The `-o` option produces output similar to the following:

CODE EXAMPLE 0-2 Sample `-o` Option Output

```
# ./jass-execute -o jass-output.txt -d secure.driver
[NOTE] Executing driver, secure.driver
[NOTE] Recording output to jass-output.txt
```

Quiet Option

The `-q` option disables Solaris Security Toolkit output to standard input output (stdio) stream during a hardening run.

This option has no effect on the logs kept in the `JASS_REPOSITORY` directory. Similar to the `-o` option, this option is particularly helpful when running the Solaris Security Toolkit software through a cron job or over slow network connections.

This option can be used with either the `-d`, `-u`, or `-a` options.

The `-q` option produces output similar to the following:

CODE EXAMPLE 0-3 Sample `-q` Option Output

```
# ./jass-execute -q -d secure.driver
[NOTE] Executing driver, secure.driver
```

Verbosity Option

The `-V` option specifies the verbosity level for an audit run. This option is only available for auditing. Verbosity levels provide a highly flexible way of displaying the results of an audit run. For example, if you have 100 machines to audit, you may want to limit the output to a single line for each machine to simply determine which machines pass or fail. Then, for the machines that fail, you might want to run an audit that produces expanded output, to focus on the problem areas.

The five verbosity levels (0 through 4) are controlled by the `-V` option. Each incremental level provides additional detail that you can use to more fully understand which checks are passing and which are failing. TABLE 0-2 describes the verbosity levels.

TABLE 0-2 Audit Verbosity Levels

Level	Output
0	Single line indicating pass or fail.
1	For each script, a single line indicating pass or fail. One grand total score line below all the script lines.
2	For each script, provides results of all checks.
3	Multiple lines providing full output, including banner and header messages.
4	Multiple lines (all data provided from level 3) plus all entries that are generated by the `logDebug` logging function. This level is for debugging.

Note – The default verbosity level for the `jass-execute -V` command is 3.

For complete descriptions of the verbosity levels, refer to Chapter 13, “JASS_VERBOSITY” on page 320.

Banners and Messages Output

You can configure the Solaris Security Toolkit audit option to report or omit banners and messages. The `JASS_LOG_BANNER` variable cannot be used with verbosity levels 0-2. These output options apply to verbosity levels 3 and 4. For example, you might want to eliminate pass messages (`JASS_LOG_SUCCESS` variable) from the output so you can report and focus only on fail messages (`JASS_LOG_FAILURE` variable).

TABLE 0-3 lists the banners and messages that you can control through logging variables. (For detailed information about logging variables, refer to Chapter 13.) If the logging variable is set to `0`, then no output is generated for messages of that type. Conversely, if the logging variable is set to `1`, then messages are displayed. The default action for each of these variables is to display the output. TABLE 0-3 describes the logging variables.

TABLE 0-3 Displaying Banners and Messages in Audit Output

Logging Variable	Log Prefix	Description
`JASS_LOG_BANNER`	All Banner Output	This parameter controls the display of banner messages. These messages are usually surrounded by separators comprised of either equal sign (“=”) or dash (“-”) characters.
`JASS_LOG_ERROR`	`[ERR]`	This parameter controls the display of error messages. If set to 0, no error messages will be generated.
`JASS_LOG_FAILURE`	`[FAIL]`	This parameter controls the display of failure messages. If set to 0, no failure messages will be generated.

TABLE 0-3 Displaying Banners and Messages in Audit Output *(Continued)*

Logging Variable	Log Prefix	Description
JASS_LOG_NOTICE	[NOTE]	This parameter controls the display of notice messages. If set to 0, no notice messages will be generated.
JASS_LOG_SUCCESS	[PASS]	This parameter controls the display of success or passing status messages. If set to 0, no success messages will be generated.
JASS_LOG_WARNING	[WARN]	This parameter controls the display of warning messages. If set to 0, no warning messages will be generated.

Using these options is very useful when you only need to view specific messages. By setting these options, you can minimize output, yet still focus on areas you deem critical. For example, by setting all logging variables to 0 except for JASS_LOG_FAILURE (leave it at the default of 1), the audit reports only on failures generated by the `logFailure` function.

CODE EXAMPLE 0-4 Sample Output of Reporting Only Audit Failures

```
# JASS_LOG_FAILURE=1
# export JASS_LOG_FAILURE
[setting of other parameters to 0 omitted]
# ./jass-execute -a secure.driver -V 2
update-at-deny        [FAIL] User test is not listed in
   /etc/cron.d/at.deny.
update-at-deny        [FAIL] Audit Check Total : 1 Error(s)
update-inetd-conf     [FAIL] Service ftp is enabled in
   /etc/inet/inetd.conf.
update-inetd-conf     [FAIL] Service telnet is enabled in
   /etc/inet/inetd.conf.
update-inetd-conf     [FAIL] Service rstatd is enabled in
   /etc/inet/inetd.conf.
update-inetd-conf     [FAIL] Audit Check Total : 3 Error(s)
```

Host Name, Script Name, and Timestamp Output

You can configure the Solaris Security Toolkit audit option to include host name, script name, and timestamp information for verbosity levels 0-2. For example, if you have many machines to audit, you may want to be able to sort the output by host name, script name, or timestamp. TABLE 0-4 lists the variables.

TABLE 0-4 Displaying Host Name, Script Name, and Timestamp Audit Output

Variable Name	Variable Description
JASS_DISPLAY_HOSTNAME	Setting this parameter to 1 causes the Solaris Security Toolkit software to prepend each log entry with the host name of the system. This information is based on the JASS_HOSTNAME parameter. By default, this parameter is empty, so the Toolkit will not display this information.
JASS_DISPLAY_SCRIPTNAME	By default, this parameter is set to 1, so the Solaris Security Toolkit software prepends each log entry with the name of the audit script currently being run. Setting this parameter to any other value causes the Toolkit to not display this information.
JASS_DISPLAY_TIMESTAMP	Setting this parameter to 1 causes the Solaris Security Toolkit software to prepend each log entry with the timestamp associated with the audit run. This information is based on the JASS_TIMESTAMP parameter. By default, this parameter is empty, so the software does not display this information.

By configuring the Solaris Security Toolkit software to prepend host, script, and timestamp information, you can combine many runs from either a single system or group of systems and sort them based on the key data. You can use the information to look for problems that span several systems or that are symptomatic of deployment processes. For example, using the information in this way, an administrator can tell if every system build using a given process always has the same failed checks.

For example, by setting the JASS_DISPLAY_TIMESTAMP parameter to 1 and setting the JASS_DISPLAY_SCRIPTNAME value at 0, output similar to the following would be generated.

CODE EXAMPLE 0-5 Sample Output of Auditing Log Entries

```
# JASS_DISPLAY_SCRIPTNAME=0
# JASS_DISPLAY_TIMESTAMP=1
# export JASS_DISPLAY_SCRIPTNAME JASS_DISPLAY_TIMESTAMP
# ./jass-execute -a secure.driver -V 2
20030101233525 [FAIL] User test is not listed in
   /etc/cron.d/at.deny.
20030101233525 [FAIL] Audit Check Total : 1 Error(s)
20030101233525 [FAIL] Service ftp is enabled in
   /etc/inet/inetd.conf.
20030101233525 [FAIL] Service telnet is enabled in
   /etc/inet/inetd.conf.
20030101233525 [FAIL] Service rstatd is enabled in
   /etc/inet/inetd.conf.
20030101233525 [FAIL] Audit Check Total : 3 Error(s)
```

Performing a Security Audit

Performing a security assessment periodically on your systems provides a benchmark of how closely the security matches the security profile you implemented. The most common scenario for performing security assessments is as a security maintenance task sometime after hardening new installations. We designed the security assessment option so that you simply execute the same hardening driver(s) that you used to harden the system, but that now you use the `-a` option to check the current state compared to the security profile implemented during hardening. This design eliminates complexity and provides flexibility. For example, when you update your security profile, subsequent security assessments use the updated security profile.

Another possible scenario is that you are responsible for securing systems that are already deployed, and before you harden them, you want to perform a security assessment. In this scenario, you would define your own security profile, customize a Solaris Security Toolkit security profile template, or use one of the security profile templates as is.

▼ To Perform a Security Audit

Before performing an audit, you need to define or choose a security profile. For more information, refer to “Preparing to Audit Security” on page 5.

Caution – If you are performing a security assessment on a deployed system that you did not harden previously, we recommend that you first back up the machine and reboot it to verify that it is in a known, working, and consistent configuration. Any errors or warnings detected during this preliminary reboot should be corrected or noted before proceeding with security assessment.

1. **Choose the security profile (hardening driver) that you want to use**:
   - If you hardened the system previously, use the same security profile.

     For example, `secure.driver`.
   - If you have not hardened the system, use one of the standard security profiles or your own.

     For example, `secure.driver or abccorp-secure.driver`.

   For a complete and up-to-date listing of available drivers, download the most recent version of the Solaris Security Toolkit software from the following web site:

   ```
   http://www.sun.com/security/jass
   ```

   Refer to Chapter 10 for information about standard and product-specific drivers. For the most current listing of drivers, refer to the Drivers directory.
2. **Determine the command line options you want and how you want to control the output. (Refer to “Using Options and Controlling Audit Output” on page 6.)**
3. **Enter the** `jass-execute -a` **command, the name of the security profile, and the options you want.**

The following is a sample audit run using the `sunfire_15k_sc-secure.driver`.

CODE EXAMPLE 0-6 Sample Output of Audit Run

```
# ./jass-execute -a sunfire_15k_sc-secure.driver
[NOTE] Executing driver, sunfire_15k_sc-secure.driver

[...]

================================================================
sunfire_15k_sc-secure.driver: Audit script: enable-rfc1948.aud
================================================================

#----------------------------------------------------------------
# RFC 1948 Sequence Number Generation
#
# Rationale for Audit:
#
# The purpose of this script is to audit that the system is
# configured and is in fact using RFC 1948 for its TCP sequence
# number generation algorithm (unique-per-connection ID). This is
# configured by setting the 'TCP_STRONG_ISS' parameter to '2' in
# the /etc/default/inetinit file.
#
# Determination of Compliance:
#
[...]
#----------------------------------------------------------------

[PASS] TCP_STRONG_ISS is set to '2' in /etc/default/inetinit.
[PASS] System is running with tcp_strong_iss=2.

# The following is the vulnerability total for this audit script.

[PASS] Audit Check Total : 0 Error(s)

================================================================

# The following is the vulnerability total for this driver profile.

[PASS] Driver Total : 0 Error(s)

================================================================
sunfire_15k_sc-secure.driver: Driver finished.
================================================================

[PASS] Grand Total : 0 Error(s)
```

When an audit run is initiated, the Solaris Security Toolkit software accesses files from the `JASS_HOME_DIR/Audit` directory. Although the files in both the `JASS_HOME_DIR/Audit` and `JASS_HOME_DIR/Finish` directories share the same base file names, they have different file name suffixes. The `driver.run` script automatically translates the finish scripts defined by the `JASS_SCRIPTS` variable into audit scripts, by changing their suffixes from `.fin` to `.aud`.

The audit run starts and initializes the state of the Solaris Security Toolkit software. Each driver that is accessed during the run evaluates the state of all of its file templates and audit scripts. Each check results in a state of success or failure, represented by a vulnerability value of either zero or nonzero, respectively. In most cases, failure is represented by a number 1. Each script that is run produces a total security score, based on the total vulnerability value of each check contained within a script. Furthermore, the total vulnerability value result for each driver is displayed at the completion of a driver's assessment. Lastly, a grand total of all scores is presented at the end of the run.

The security assessment option provides a comprehensive view of the state of a system at the time the assessment run is initiated. The Solaris Security Toolkit software checks the stored state of the system by inspecting configuration files and checks the running state of the system by inspecting process table information, device driver information, etc. The Solaris Security Toolkit software checks not only for the existence of each file or service, but it checks if the software associated with a service is installed, configured, enabled, and running. This holistic approach yields an accurate snapshot of the current state of a system.

About the Authors

Alex Noordergraaf

Alex Noordergraaf has over 10 years experience in the areas of computer and network security. As the Security Architect of the Enterprise Server Products (ESP) group at Sun Microsystems, he is responsible for providing technical leadership to define the security of Sun's next generation servers while addressing security for current products. He is the driving force behind the very popular freeware Solaris Security Toolkit. Prior to his role in ESP, he was a Senior Staff Engineer in the Enterprise Engineering (EE) group of Sun Microsystems, where he developed, documented, and published security best practices through the Sun BluePrints program. Published topics include: Sun Fire Midframe 15K system security, secure N-tier environments, Solaris OE minimization, Solaris OE network settings, and Solaris OE security. He has co-authored two Sun BluePrint books *Jumpstart Technology - Effective Use in the Solaris Operating Environment* and *Enterprise Security Solaris Operating Environment, Security Journal*.

Prior to his role in EE, 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. He developed and delivered an enterprise security assessment methodology and training curriculum to be used worldwide by SunPS. His customers included major telecommunication firms, financial institutions, ISPs, and ASPs. Before joining Sun, Alex was an independent contractor specializing in network security. His clients included BTG, Inc. and Thinking Machines Corporation.

Glenn Brunette

Glenn Brunette is a Sun Principal Engineer with over a decade of experience in information security. Glenn works in the Sun Professional Services division as the Americas Lead Security Architect. In this role, he is responsible for the development and execution of the region's security services strategy. He works with teams throughout the Americas and the world to improve the quality and security of services delivered to Sun's customers.

Previously, Glenn worked in the North East and Financial Services Areas developing and delivering a wide array of tailored security solutions supporting the lifecycle of assessment, architecture, implementation, and management. His customers have

included major financial services firms, service providers, and life sciences and government organizations. In addition to contract services, Glenn works closely with teams across Sun on the development and delivery of security strategy, methodologies, best practices, training, and tools. Glenn is a co-founder of the very popular freeware Solaris Security Toolkit software. Glenn is a Certified Information Systems Security Professional (CISSP) and has been trained in the National Security Agency's INFOSEC Assessment Methodology (IAM).

Ordering Sun Documents

The SunDocs℠ program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is `http://docs.sun.com/`

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at: `http://www.sun.com/blueprints/online.html`