SUN BLUEPRINTS [illegible] APPROVED

The Solaris™ Security Toolkit - Installation, Configuration and Usage Guide

Updated for Toolkit version 0.3

By Alex Noordergraaf - Enterprise Engineering and Glenn Brunette - Sun Professional Services

Sun BluePrints™ OnLine - June 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.: 816-1384-10
Revision 1.1, 04/10/02
Edition: June 2001

The Solaris™ Security Toolkit - Installation, Configuration, and Usage Guide

Updated for Toolkit version 0.3

Overview

This is the third article in a four-part series discussing the Solaris™ Security Toolkit as a mechanism of securing systems using the Solaris™ Operating Environment (Solaris OE).

Advanced configuration and usage options available in the Toolkit are explored in this article, in addition to the philosophy and problems that have driven Toolkit development. The goal of this article is to help Toolkit users effectively utilize the capabilities of the Toolkit.

First, the philosophy behind the Toolkit and the problems which drove the initial development of the Toolkit are described. Solaris OE versions supported by the Toolkit and Toolkit support are also discussed.

Next, this article describes the configuration options available in the Toolkit. These options are significantly enhanced in version 0.3 of the Toolkit. The goal of these options is to minimize Toolkit code changes required, while customizing the Toolkit to a specific security policy.

Several new command line options to `jass-execute` have been added to this version of the Toolkit. Each of these options, and particularly the new undo feature, are discussed and sample output provided.

Lastly, the newly added `make-dist` script is described. This feature has been added to allow Toolkit users a simple mechanism by which custom-built Toolkit packages can be created for easy distribution within their organizations.

Update

This Sun BluePrints™ OnLine article has been updated to reflect changes in the newly released version (0.3) of the Solaris Security Toolkit for the Solaris OE. The documentation for this release of the Toolkit 0.3 has been re-written into four parts:

- *Quick Start* focuses on the minimal set of required information to get the Toolkit up and running. Setup and configuration requirements for the Toolkit are quite different, depending on whether it is being run in standalone or JumpStart™ modes, so this article will discuss each method.
- *Release Notes* discusses the changes and enhancements included in the new release.
- *Installation, Configuration, and Usage Guide* focuses on installation, configuration, and usage information not contained in the Quick Start guide.
- *Internals* focuses on the actual components of the Toolkit. Each of the internal scripts are individually discussed.

Philosophy

The goal of the Toolkit is to automate and simplify building secured Solaris OE systems. The Toolkit focuses on Solaris OE security modifications to harden and minimize a system. Hardening is the modification of Solaris OE configurations to improve the security of the system. Minimization is the removal of unnecessary Solaris OE packages from the system. This removal reduces the number of components to be patched and made secure, which, in turn, has the potential to reduce entry points available to a possible intruder.

Note – Configuration modifications for performance enhancements and software configuration are not addressed by the Toolkit.

The Toolkit was designed to harden systems in one of two modes; standalone or jumpstart.

Standalone Mode

The Toolkit has been designed to be run directly from a Solaris OE shell prompt in standalone mode. This standalone mode allows the Toolkit to be used on systems that require security modifications or updates, but which cannot be taken out of service to re-install the OS from scratch. Ideally, however, systems to be secured should be re-installed from scratch.

Standalone mode is particularly useful when re-hardening a system after patches have been installed. The Toolkit may be run any number of times on a system with no ill effects. Patches may overwrite or modify files the Toolkit has also modified; by re-running the Toolkit, any security modifications undone by the patch installation can be reimplemented. In production environments, patches should always be staged in test and development environments before installation.

JumpStart Technology Mode

Systems should be hardened during installation. The Toolkit can be used to harden systems during installation through the use of JumpStart technology. JumpStart technology, which is Sun's network-based Solaris OE installation mechanism, can also run scripts during the installation process, which is used to execute the Toolkit scripts. Readers not familiar with jumpstart technology are referred to the Sun BluePrints OnLine article titled *Building a JumpStart™ Infrastructure* (April 2001), for detailed instructions on how to set up a JumpStart server and environment. Refer to the Bibliography for the location of this article. That article is an excerpt from the forthcoming Sun BluePrints book titled *JumpStart™ Technology: Effective use in the Solaris™ Operating Environment*, by John S. Howard and Alex Noordergraaf. This book is scheduled for publication by Prentice Hall in the late Summer of 2001 (ISBN# 0-13-062154-4).

The Toolkit was built with a modular framework. Customers with existing JumpStart installations will benefit from Toolkit's ability to integrate into existing JumpStart architecture. For customers not currently using the JumpStart product, the flexibility of the Toolkit's framework will allow them an efficient beginning.

Problem

The time-to-market time frame for many businesses is being eroded at breakneck speed. This is especially true in today's internet driven economy—consequently, there is less time to perform all tasks critical for the security of the infrastructure.

Manually dealing with security issues for each server on an individual basis is extremely time consuming, and does not scale in the enterprise. Tools have been developed to address these issues in both the freeware and commercial arenas; however, many of the tools can only be used at the individual server level, and generally have to be run manually following the installation and configuration of a server.

A process has been needed that will automatically install the operating system and configure all necessary security functions. JumpStart technology—available for the Solaris OE product since version 2.1—is currently used by many organizations to automate OS installation and configuration. However, not all organizations are using the JumpStart framework to optimize the security features of their installations. This Toolkit has been developed, in part, to assist organizations that currently use the JumpStart product to enhance their installations, and to assist organizations just beginning to use the JumpStart product.

An important justification for this framework is improved server baseline security. By having the process and technology available, it will be possible to ensure that every server has the necessary modifications.

An automated and non-interactive installation process has additional benefits. By using the Toolkit, a process can be developed that captures and communicates knowledge. This process is critical when training new staff, as well as for capturing updates and documenting information for other staff members. The JumpStart environment can be used to help implement updates to the environment; either by re-building the entire system from scratch with new updates, or by installing the new software directly onto the system. Other benefits include the simplification of system reconstruction due to major hardware failures and replacements.

Supported Versions

The current release of the Toolkit works with Solaris OE versions 2.5.1, 2.6, 7, and 8. The Toolkit scripts will automatically detect which version of the Solaris OE software is installed, and only run tasks appropriate to that version.

Support Forums

With the release of Toolkit version 0.3 all bug reports, questions, suggestions, and feedback to the Toolkit developers are being migrated to the Sun™ SupportForum web site at `http://supportforum.sun.com/salerts`. In the Sun Alerts and

Security Support section, there is a forum called `jass Security Toolkit Discussions`, which should be used for all Toolkit related questions, comments, and suggestions. As always, feedback on how the Toolkit works and general words of encouragement to the developers are appreciated.

Installation and Basic Toolkit Configuration

Toolkit installation and basic configuration issues are discussed in *The Solaris™ Security Toolkit - Quick Start: Updated for Toolkit version 0.3* Sun BluePrints OnLine article. Refer to the Bibliography or the Toolkit `Documentation` directory for the appropriate PDF files.

Toolkit Architecture

The main components of the architecture consist of the following directories:

- `Documentation`
- `Drivers`
- `Files`
- `Finish`
- `OS`
- `Packages`
- `Patches`
- `Profiles`
- `Sysidcfg`

The contents of these directories are discussed in the Sun BluePrints OnLine article titled *The Solaris™ Security Toolkit - Internals: Updated for Toolkit version 0.3*. This article focuses on the internal components of the Toolkit, such as directory structures and their contents. Refer to the Bibliography for the location of this article. It is also available in the `Documentation` directory of Toolkit version 0.3.

Advanced Toolkit Configuration

The Toolkit architecture includes configuration information to enable driver and finish scripts to be used in different environments, while not modifying the actual finish scripts themselves. All variables used in the finish scripts are maintained in a set of configuration files—these configuration files are imported by driver scripts, which make the variables available to the finish scripts as they are called by the driver.

The Toolkit has three main configuration files, all of which are stored in the `Drivers` directory: `driver.init`, `finish.init`, and `user.init`.

`driver.init`

This script contains variables that define aspects of the Toolkit framework and overall operation.

Note – The `driver.init` script should not be altered, as it will be overwritten in subsequent versions of the Toolkit. All user modifications and variable overrides should occur in the `user.init` script.

This script contains the following variables:

- `JASS_FILES_DIR`
- `JASS_FINISH_DIR`
- `JASS_HOME_DIR`
- `JASS_HOSTNAME`
- `JASS_PACKAGE_DIR`
- `JASS_PACKAGE_MOUNT`
- `JASS_PATCH_DIR`
- `JASS_PATCH_MOUNT`
- `JASS_PKG`
- `JASS_REPOSITORY`
- `JASS_ROOT_DIR`
- `JASS_RUN_FINISH_LIST`
- `JASS_RUN_INSTALL_LOG`
- `JASS_RUN_MANIFEST`
- `JASS_RUN_UNDO_LOG`
- `JASS_RUN_VERSION`
- `JASS_SAVE_BACKUP`
- `JASS_STANDALONE`
- `JASS_SUFFIX`
- `JASS_TIMESTAMP`
- `JASS_UNAME`
- `JASS_USER_DIR`
- `JASS_VERSION`

Each of these environment variables is examined in greater detail:

JASS_FILES_DIR

This variable points to the location of the `Files` directory under `JASS_HOME_DIR`. This directory contains files which can be copied to the client.

Any files to be copied are specified in the `JASS_FILES` variable—these will be copied to the client during installation. The `JASS_FILES` variable is set by individual drivers and not in the configuration file. There are several methods available for copying files using this variable which will be covered in *The Solaris™ Security Toolkit - Internals: Updated for Toolkit version 0.3* Sun BluePrints OnLine article.

The `JASS_FILES_DIR` variable should not normally require modification.

JASS_FINISH_DIR

The convention used by the Toolkit is to store all finish scripts in the `Finish` directory. However, for flexibility, the `JASS_FINISH_DIR` environment variable has been included for those organizations that require finish scripts to be stored in different locations.

This variable should not normally require modification.

JASS_HOME_DIR

This variable defines the location of the Toolkit source tree. In JumpStart mode, the JumpStart variable `SI_CONFIG_DIR` will be used to set `JASS_HOME_DIR`. In standalone mode, it will be set by the `jass-execute` script which is included in the base directory of the Toolkit.

Normally this variable should not require modification by the user, except when the Toolkit is installed into a subdirectory of a pre-existing JumpStart installation. For these cases, the path to the Toolkit source should be appended to `SI_CONFIG_DIR`, as in `SI_CONFIG_DIR/jass-0.3`. For more information refer to the Frequently Asked Questions section of *The Solaris™ Security Toolkit - Quick Start: Updated for Toolkit version 0.3* Sun BluePrints OnLine article for specific code modifications.

JASS_HOSTNAME

Contains the host name of the system on which the Toolkit is being installed and is set during a Toolkit run through the use of the Solaris OE `uname -n` command.

This variable should not be changed.

JASS_PACKAGE_DIR

The JASS_PACKAGE_DIR variable specifies the directory where the packages directory will be mounted during a JumpStart installation. Normally, the JASS_PACKAGE_DIR variable will not require modification, because it is a transient mount-point used only during the JumpStart installation.

This variable should not normally be modified through the user.init script.

JASS_PATCH_DIR

The JASS_PATCH_DIR variable specifies the directory where the Patch directory will be mounted during a JumpStart installation. Normally, the JASS_PATCH_DIR variable will not usually require modification, because it is a transient mount-point used only during JumpStart installations.

This variable should not normally require modification through the user.init script.

JASS_PKG

This variable defines the package name of the Toolkit. By default this is defined as SUNWjass.

This variable should not be changed.

JASS_REPOSITORY

This variable is used as part of the execution log and undo modules. The path specified by JASS_REPOSITORY will be used to define the directory in which the required run information is stored. This will facilitate the determination of scripts run, in addition to listing the files installed and modified for any given run.

This variable is dynamically altered during the execution of the Toolkit. Any values assigned to this variable in any of the init files will be overwritten.

JASS_ROOT_DIR

This variable defines the root directory of the file system. For JumpStart installations, this will always be `/a`. For standalone Toolkit executions, this variable should be set to `/` or the root directory of the system.

Toolkit version 0.2 and above automates this in the `jass-execute` script, so manual modification is no longer required.

JASS_RUN_FINISH_LIST

This variable is used as part of the execution log. The absolute path and filename specified by `JASS_RUN_FINISH_LIST` is used to store a listing of all of the finish scripts executed during a Toolkit run.

This variable should not be changed.

JASS_RUN_INSTALL_LOG

The absolute path and filename specified by `JASS_RUN_INSTALL_LOG` is used to define the location of the output of a Toolkit run. This facilitates the determination of scripts run, in addition to listing files installed and modified for any given run. In addition, any errors or warnings that may have been generated will be stored in this file. The information stored in this file is equivalent to the output displayed to standard output during a standalone Toolkit run.

This variable should not be changed.

JASS_RUN_MANIFEST

This variable is used as part of the execution log and undo modules. The path specified by `JASS_RUN_MANIFEST` is used to define where the `MANIFEST` of a Toolkit run is kept. This `MANIFEST` is used by the undo feature to determine what files must be moved, and in what order, to restore a system to a previous configuration.

This variable should not be changed.

JASS_RUN_UNDO_LOG

This variable is used as part of the undo modules. The path specified by `JASS_RUN_UNDO_LOG` is used to define the absolute path and filename of the file which will contain the output of a Toolkit undo run. This facilities the determination of operations were done during a Toolkit run in undo mode.

This variable should not be changed.

JASS_RUN_VERSION

This variable defines the absolute path to the file containing the version and run information for a particular run of the Toolkit.

This variable should not be changed.

JASS_SAVE_BACKUP

This variable controls the creation of backup files during Toolkit execution. The default value of this variable is 1, which causes the Toolkit to create a backup copy of any file modified on client. If the value is changed to 0, then all backup copies will be removed from the system.

This variable can be modified if backup copies of files should not be created. Modifications to this variable should be made in the `user.init` script. The value to which the variable is set in the `user.init` script will overwrite any previously set values.

Note – The Toolkit undo feature will be unavailable if `JASS_SAVE_BACKUP` is defined as 0.

JASS_STANDALONE

This variable is used to control whether the Toolkit runs in standalone or JumpStart mode. When set to 1, the Toolkit runs in standalone, while the value of 0 is used for JumpStart mode. The `jass-execute` script will set it for standalone mode execution. The default value of 0 is correct for JumpStart installations.

This variable is automatically set in the `jass-execute` script. Manual modification is not required.

JASS_SUFFIX

This variable is used by the Toolkit to determine which suffixes must be appended onto backup copies of files. By default this is set to `JASS.<timestamp>`. During a Toolkit run the timestamp used will change to both reflect the time a file was created and to guarantee that all backup file names are unique.

This variable is dynamically altered during the execution of the Toolkit. Any values assigned to this variable in any of the `init` files will be overwritten.

JASS_TIMESTAMP

The value of this variable is used to create the `/var/opt/SUNWjass/run/JASS_TIMESTAMP` directory, which will contain the logs and manifest information for each individual run of the Toolkit.

This variable is set during the Toolkit run and should not be set by the user.

JASS_UNAME

This variable is used as a global environment variable specifying the OS version of the client being built. This variable is set by the `driver.init` script through the use of the `uname -r` command and exported so all other scripts can access it.

This variable is set during the Toolkit run and should not be set by the user.

JASS_USER_DIR

This variable specifies the location of the Toolkit configuration files `user.init` and `user.run`. By default, these files are stored in the `Drivers` directory. Any custom modifications to the Toolkit required should be implemented in these files to minimize the impact of Toolkit upgrades in the future.

This variable can be changed. Modifications to this variable should be made in the `user.init` script. The value to which the variable is set in the `user.init` script will overwrite any previously set values.

JASS_VERSION

This variable defines the version of the Toolkit being applied to the system. For this release of the Toolkit, this variable is set to 0.3.

This variable should not be changed.

finish.init

This file contains variables that define the behavior of the individual finish scripts. There are two factors that contribute how a system will be hardened: (1) the driver script selected contains the list of finish scripts to execute and files to install, and (2) the `finish.init` script defines how the executed finish scripts will act.

Note – The `finish.init` script should not be altered as it will be overwritten in subsequent versions of the Toolkit. All user modifications and variable overrides should occur in the `user.init` script.

This script contains the following variables:

- `JASS_ACCT_DISABLE`
- `JASS_ACCT_REMOVE`
- `JASS_AGING_MAXWEEKS`
- `JASS_AGING_MINWEEKS`
- `JASS_AGING_WARNWEEKS`
- `JASS_AT_ALLOW`
- `JASS_AT_DENY`
- `JASS_CPR_MGT_USER`
- `JASS_CRON_ALLOW`
- `JASS_CRON_DENY`
- `JASS_CRON_LOG_SIZE`
- `JASS_FTPD_UMASK`
- `JASS_FTPUSERS`
- `JASS_KILL_SCRIPT_DISABLE`
- `JASS_LOGIN_RETRIES`
- `JASS_PASSWD`
- `JASS_PASS_LENGTH`
- `JASS_POWER_MGT_USER`
- `JASS_RHOSTS_FILE`
- `JASS_ROOT_PASSWORD`
- `JASS_SADMIND_OPTIONS`
- `JASS_SENDMAIL_MODE`
- `JASS_SGID_FILE`
- `JASS_SHELLS`
- `JASS_SUID_FILE`
- `JASS_SUSPEND_PERMS`
- `JASS_SVCS_DISABLE`
- `JASS_TMPFS_SIZE`
- `JASS_UMASK`
- `JASS_UNOWNED_FILE`
- `JASS_WRITEABLE_FILE`

Each of these environment variables is examined in greater detail:

JASS_ACCT_DISABLE

This variable contains a (possibly empty) list of users to be disabled as part of the `disable-system-accounts.fin` finish script. By default, all administrative users with the exception of root and sys are disabled. Only those accounts shipped by default with the Solaris OE that have user identification numbers less than 100 or greater than 60000 will be affected.

JASS_ACCT_REMOVE

This variable contains a (possibly empty) list of users to be removed from the system as part of the `remove-uneeded-accounts.fin` finish script. By default, the accounts `listen`, `nobody4`, and `smtp` are removed from the system.

JASS_AGING_MAXWEEKS

This variable contains a numeric value specifying the maximum number of weeks a password remains valid before it must be changed by the user. The default value for this variable is 8. This variable is used in the `set-user-password-reqs.fin` finish script.

JASS_AGING_MINWEEKS

This variable contains a numeric value specifying the minimum number of weeks that must pass before a user can change their password. This variable is used in the `set-user-password-reqs.fin` finish script and has a default value of 1.

JASS_AGING_WARNWEEKS

This variable contains a numeric value specifying the number of weeks before a password expiration that the user is warned. This variable is used in the `set-user-password-reqs.fin` finish script. The default value for this variable is 1.

JASS_AT_ALLOW

This variable contains a (possibly empty) list of users to be added to the `/etc/cron.d/at.allow` file. A user will not be added to this file if it already exists in `/etc/cron.d/at.deny` or if it does not exist in JASS_PASSWD. This variable is used by the `install-at-allow.fin` finish script and by default contains no users.

JASS_AT_DENY

This variable contains a (possibly empty) list of users to be added to the /etc/cron.d/at.deny file. A user will not be added to this file if it already exists in /etc/cron.d/at.allow or if it does not exist in JASS_PASSWD. By default, all users in the password file are assigned to this variable. This variable is used by the update-at-deny.fin finish script.

JASS_CPR_MGT_USER

This variable contains a string value that will be used to define what users will be permitted to perform checkpoint resume functions. This default value for this variable is "-" which indicates that only the root account will be able to perform these management functions. For more information, refer to the /etc/default/power file. This variable is used in the set-power-restrictions.fin finish script.

JASS_CRON_ALLOW

This variable contains a (possibly empty) list of users to be added to the /etc/cron.d/cron.allow file. Note that a user will not be added to this file if it already exists in /etc/cron.d/cron.deny or if it does not exist in the /etc/password file. By default, this variable only contains the root account. This variable is used by the update-cron-allow.fin finish script.

JASS_CRON_DENY

This variable contains a (possibly empty) list of users to be added to the /etc/cron.d/cron.deny file. Note that a user will not be added to this file if it already exists in /etc/cron.d/cron.allow or if it does not exist in the password file. By default, this variable is populated with users whose identification numbers are less than 100 or greater than 60000. These ranges are traditionally reserved for administrative accounts. This variable is used by the update-cron-allow.fin finish script.

JASS_CRON_LOG_SIZE

This variable contains a numeric value representing the maximum size (in blocks) of /var/cron/log file. If the file exceeds this maximum limit, it will be moved to /var/cron/olog by the /etc/cron.d/logchecker script, which is executed by `cron` for the `root` user. The default value for this script was originally 1024 (0.5MB), but has been changed to 20480 (10.0MB). This variable is used by the `update-cron-log-size.fin` finish script.

JASS_FTPD_UMASK

This variable contains a numeric (octal) value for the default file creation mask used by the `in.ftpd(1M)` daemon. This variable is used in the `set-ftpd-umask.fin` finish script and has a default value of 022.

JASS_FTPUSERS

This variable contains a (possibly empty) list of users to be added to the `/etc/ftpusers` file. By default, this variable is populated with users whose identification numbers are less than 100 or greater than 60000. These ranges are traditionally reserved for administrative accounts. This variable is used by the `install-ftpusers.fin` finish script.

JASS_KILL_SCRIPT_DISABLE

This variable contains a boolean value that determines whether the kill run control scripts (for a given service or finish script) will be disabled. The start run control scripts are always disabled. Some administrators prefer to have the kill scripts left in place so that any services that may be started manually will be properly terminated during a system shutdown or reboot. The default value of `1` indicates that kill scripts will be disabled.

JASS_LOGIN_RETRIES

This variable contains a numeric value specifying the number of consecutive failed logins that can occur for a user before the login process logs the failure and terminates the connection. This variable is used in the `set-login-retries.fin` finish script and has a default value of 3.

JASS_PASSWD

This variable contains a filename value that specifies the location of the password file on the system the Toolkit run is being performed on. This variable, and the password file it points to, is used by many of the variables defined in the `finish.init` file. The default value of this variable is set to `$JASS_ROOT_DIR/etc/password`.

JASS_PASS_LENGTH

This variable contains a numeric value specifying the minimum length of a user password. The valid range for this variable is between 1 and 8. This variable is used in the `set-user-password-reqs.fin` finish script and has a default value of 8.

JASS_POWER_MGT_USER

This variable contains a string value that will be used to define what users will be permitted to perform power management functions. This default value for this variable is "-" which indicates that only the `root` account will be able to perform power management functions. For more information, refer to the `/etc/default/power` file. This variable is used in the `set-power-restrictions.fin` finish script.

JASS_RHOSTS_FILE

This variable specifies where the `print-rhosts.fin` finish script sends its output. If the variable is not defined or has a null value, the output is sent to standard output. The default configuration of the Toolkit is to not define `JASS_RHOSTS_FILE` and have the output directed to standard output.

JASS_ROOT_PASSWD

This variable specifies the encrypted root password used by the `set-root-password.fin` finish script. This will only be executed when using the Toolkit in JumpStart mode. The `set-root-password.fin` finish script does not run when the Toolkit is run in standalone mode. The default root password supplied with the Toolkit is ‘`t00lk1t`’.

JASS_SADMIND_OPTIONS

This variable contains a string value specifying options used with the `sadmind` daemon executed from the `inetd` process. By default, a value of `-S 2` is used to enable strong authentication (`AUTH_DES`) when communicating with clients. This variable is used in the `install-sadmind-options.fin` finish script.

JASS_SENDMAIL_MODE

This variable contains a string value specifying options used by `/usr/lib/sendmail` for its mode. For example, if the daemon should accept incoming SMTP connections, then the string `-bd` should be used. If the daemon should only perform queue processing, then the empty string ("") should be used. This variable is used in the `disable-sendmail.fin` finish script. Note that this variable is currently only used to enable or disable daemon mode operation on Solaris 8 OE systems. The default `JASS_SENDMAIL_MODE` used in Toolkit version 0.3 is "". For more information on this sendmail feature, refer to the *Solaris™ Operating Environment Security: Updated for Solaris 8 Operating Environment* (April 2001) Sun BluePrints OnLine article. This article is available in the Toolkit `Documentation` directory or through the URL listed in the Bibliography.

JASS_SGID_FILE

This variable specifies where the `print-sgid-files.fin` finish script sends its output. If the variable is not defined or has a null value, then the output is sent to standard output. The default configuration of the Toolkit is to not define `JASS_SGID_FILE` and have the output directed to standard output.

JASS_SHELLS

This variable contains a list of shells to be added to the `/etc/shells` file. The default shells for each version of the Solaris OE are defined in the `finish.init` file. This variable is used by the `install-shells.fin` finish script.

JASS_SUID_FILE

This variable specifies where the `print-suid-files.fin` finish script sends its output. If the variable is not defined or has a null value then the output is sent to standard output. The default configuration of the Toolkit is to not define `JASS_SUID_FILE` and have the output directed to standard output.

JASS_SUSPEND_PERMS

This variable contains a string value that will be used to define what users will be permitted to perform system suspend or resume functions. This default value for this variable is "-" which indicates that only the `root` account will be able to perform these management functions. For more information, refer to the `/etc/default/sys-suspend` file. This variable is used in the `set-sys-suspend-restrictions.fin` finish script.

JASS_SVCS_DISABLE

This variable can be used to simplify the removal of different services from the /etc/inet/inetd.conf file. When specified, the list of services defined in this variable will be disabled by the update-inetd-conf.fin finish script. The default list of services includes all of the entries that are provided by default with the Solaris OE.

Caution – Be certain to have either console access to the system or a non-default remote access capability, such as Secure Shell, because TELNET, RSH, and RLOGIN servers are all disabled by default.

JASS_TMPFS_SIZE

This variable contains a string value representing the amount of space allocated to the /tmp (tmpfs) file system. This value should be set large enough to handle current /tmp needs. This variable has a default value of 512-Mbytes and is used in the set-tmpfs-limit.fin finish script.

JASS_UMASK

This variable contains a numeric (octal) value to be used for both the system and user default file creation masks. This variable is used in the set-system-umask.fin and set-user-umask.fin finish scripts. The default value for this variable is 022.

JASS_UNOWNED_FILE

This variable specifies where the print-unowned-files.fin finish script sends its output. If the variable is not defined or has a null value, then the output is sent to standard output. The default configuration of the Toolkit is to not define JASS_UNOWNED_FILE and have the output directed to standard output.

JASS_WRITEABLE_FILE

This variable specifies a where the print-world-writeable-files.fin finish script sends its output. If the variable is not defined or has a null value, then the output is sent to standard output. The default configuration of the Toolkit is to not define JASS_WRITEABLE_FILE and therefore have the output directed to standard output.

user.init

This file should contain any user-defined variables. Variables defined in the driver.init and finish.init files can also be overridden if defined in this file. This allows a site to customize the Toolkit to suit their needs and specific requirements.

By default, only *these environment* variables need to be verified when moving the JumpStart environment from one site to another:

- JASS_HOME_DIR
- JASS_PACKAGE_MOUNT
- JASS_PATCH_MOUNT

The environment variable JASS_HOME_DIR is described in the driver.init section. The other two environment variables are listed in the user.init file exclusively and are described here:

JASS_PACKAGE_MOUNT

The JASS_PACKAGE_MOUNT variable identifies the location of software packages available for installation on the JumpStart server. The location must be specified by hostname or IP address and complete path to provide the NFS daemon enough information to mount the directory during installation. Since a hostname or IP address is specified in the value of the environment variable, it will *always* require modification and is therefore defined in the user.init file. This is a jumpstart specific variable, and is not used during standalone installations.

This variable requires modification for any jumpstart based installations.

JASS_PATCH_MOUNT

The JASS_PATCH_MOUNT variable specifies the JumpStart server hostname or IP address and complete path of the Patch directory; therefore, the JASS_PATCH_MOUNT variable will require modification for each site. The location must be specified by hostname or IP address and complete path to provide NFS with enough information to mount the directory during installation. Since a hostname or IP address is specified in the value of the environment variable, it will always require modification, and is therefore defined in the user.init file. This is a jumpstart specific variable, and is not used during standalone installations.

This variable requires modification for any jumpstart based installations.

Usage

JumpStart Technology Usage

Usage options of the Toolkit in JumpStart mode are discussed in the *The Solaris™ Security Toolkit - Quick Start: Updated for Toolkit version 0.3* Sun BluePrints OnLine article. In JumpStart mode, the Toolkit can really only be used in either hardening or minimization modes. Specifically, the minimization mode is only available in JumpStart mode. The Toolkit mode is controlled by the Toolkit driver inserted in the `rules` file on the JumpStart server. The following drivers are included with Toolkit version 0.3:

- `audit.driver`
- `config.driver`
- `hardening-jumpstart.driver`
- `hardening.driver`
- `install-iPlanetWS.driver`
- `secure.driver`
- `undo.driver`

Each of these drivers are discussed in *The Solaris™ Security Toolkit - Internals: Updated for Toolkit version 0.3* Sun BluePrints OnLine article.

For more information on the JumpStart technology, see the Sun BluePrints OnLine article *Building a JumpStart™ Infrastructure* (April 2001). Refer to the Bibliography or the Toolkit `Documentation` directory for the appropriate PDF files of the articles referenced.

`add-client` and `rm-client`

Two commands to simplify the additional and removal of clients from JumpStart servers have been part of the Toolkit since version 0.2. The `add_install_client` and `rm_install_client` JumpStart server commands, to add and remove JumpStart clients from the JumpStart server, can become quite lengthy. For simplicity's sake, two scripts have been included with the Toolkit; `add-client` and `rm-client`. While the usage of these commands is described in the following paragraphs, the underlying JumpStart technology is not discussed. Refer to the *Building a JumpStart™ Infrastructure* (April 2001) Sun BluePrints OnLine article for additional information on JumpStart technology.

The `add-client` script is a wrapper around the `add_install_client` script which accepts the following arguments:

Usage: `add-client client OS class server`

`client`
The resolvable hostname of the JumpStart client.

`OS`
The revision of the Solaris OE that is to be installed on the client. If no value is specified, a list of available Solaris OE versions in the `OS` directory will be provided.

`class`
The machine class of the JumpStart client. This value is in the same format as the output of the `uname -m` command.

`server`
The IP address of the JumpStart server interface for this JumpStart client. If no value is specified, a list of available options will be provided.

To add a JumpStart client called `jordan`, which is a sun4u machine, to a JumpStart server called `nomex` using Solaris 8 OE 4/01 on an interface called `nomex-jumpstart,` the following `add-client` command would be used:

```
# ./add-client jordan Solaris_8_2001-04 sun4u nomex-jumpstart
updating /etc/bootparams
```

The `rm-client` script is a wrapper around `rm_install_client` in much the same way as `add-client`:

Usage: `rm-client client`

`client`
The resolvable hostname of the JumpStart client.

To remove a JumpStart client called `jordan` the following `rm-client` command would be used:

```
# ./rm-client jordan
removing jordan from bootparams
```

Additional information on the JumpStart command being used is available in the *Building a JumpStart Infrastructure* (April 2001) Sun BluePrints OnLine article. This article is available in the Toolkit version 0.3 `Documentation` directory and is also referenced in the Bibliography.

Standalone Usage

There are several options available to the Toolkit user from `jass-execute` when using the Toolkit in standalone mode. The options available with `jass-execute` are:

Usage: `jass-execute {-d driver | -u [-n]} [-r root directory [-o output_file][-h]`

`-d driver`
The `-d` option is used to specify the *driver* script to be run in standalone mode. As described in *The Solaris™ Security Toolkit -Quick Start: Updated for Toolkit version 0.3* Sun BluePrints OnLine article, a *driver* must be specified with the `-d` option. The Toolkit prepends `Drivers/` to the name of the script added, so if the script is in the `Drivers` directory, only the script itself need be entered on the command line. The `-d` option cannot be used in conjunction with either the `-u` or `-n` options.

A `jass-execute` hardening run using the `-d` option run will have output similar to the following:

```
# ./jass-execute -d secure.driver
./jass-execute: NOTICE: Executing driver, secure.driver

================================================================
secure.driver: Driver started.
================================================================

================================================================
secure.driver: Copying personalized files.
================================================================
[...]
```

`-h`
The `-h` option is used to display the `jass-execute` help message which provides an overview of the available options.

A `jass-execute` run with the `-h` option will have output similar to the following:

```
# ./jass-execute -h
./jass-execute {-d driver | -u [-n]} [-r root directory] \
[-o output_file] [-h]
```

`-u`
The `-u` option is used to undo the modifications made during the previous Toolkit hardening runs, which could have been done either in standalone or

JumpStart modes. The undo function steps through the manifest files generated during a Toolkit run and stored in the `/var/opt/SUNWjass/runs/$JASS_TIMESTAMP` directories, and restores the backed up files to their original locations. If files were not backed up, then the undo function is not available. When performing an undo, operating files which were modified in earlier Toolkit runs will be overwritten, regardless of whether modifications have been made to them since the Toolkit hardening run was performed. Systems should be backed up before performing an undo operation if system files have been modified. The `-u` option cannot be used in conjunction with the `-d` option.

A `jass-execute` undo run will have output similar to the following:

```
# ./jass-execute -u
./jass-execute: NOTICE: Executing driver, undo.driver
Please select from one of these backups to restore to
1.  May 31, 2001 at 22:12:43 (//var/opt/SUNWjass/run/
20010531221243)
2.  May 31, 2001 at 20:10:37 (//var/opt/SUNWjass/run/
20010531201037)
3.  Restore from all runs
Choice? 2
./jass-execute: NOTICE: Restoring to previous run //var/opt/
SUNWjass/run/20010531221243

================================================================
===
undo.driver: Driver started.
================================================================
===
```

It is important to carefully select the run number when using the Toolkit undo feature. When selecting which run to restore to, note that runs are listed in reverse numerical order, by date. The system is restored to the state it was in before the selected Toolkit run was performed. In the previous example, the 22:10 run is listed before the 20:10 run.

Note – The system is restored to the state it was in before the selected Toolkit run was performed.

In the previous example, by selecting option 2, the system will be restored to the state it was in before the initial Toolkit run on May 31st at 20:10 was performed. This means that both the 22:12 and 20:10 runs will be undone to restore the system to its state prior to the 20:10 run.

`-n`

The `-n` option is only used in conjunction with the `-u` option. During a hardening run, through either standalone or JumpStart modes, the Toolkit generates a cryptographic checksum of each file modified file. These checksums are used, when the `-n` is specified, to compare against a checksum generated during the undo run. This checksum comparison will discover any files which have been modified since the Toolkit hardening run. The default behavior of the Toolkit during an undo run will be to overwrite the file, regardless of whether the checksums match or not. The `-n` option changes this behavior, and any mismatches in the checksum files will cause an error to be logged and the file will not be overwritten. This behavior can cause the system to end up in an inconsistent state after an undo run if care is not taken. The `-n` option can only be used if the `-u` option is specified.

A `jass-execute` undo run using the `-n` option will have output similar to the following:

```
# ./jass-execute -u -n
./jass-execute: NOTICE: Executing driver, undo.driver
Please select from one of these backups to restore to
1.  May 31, 2001 at 22:26:36 (//var/opt/SUNWjass/run/
20010531222636)
2.  May 31, 2001 at 20:10:37 (//var/opt/SUNWjass/run/
20010531201037)
3.  Restore from all runs
Choice?  3
./jass-execute: NOTICE: Restoring to previous run //var/opt/
SUNWjass/run/20010531222636

===============================================================
undo.driver: Driver started.
===============================================================
[...]
```

`-o output_file`

The `-o` option can be used to re-direct the console output of `jass-execute` runs to a separate file, `output_file`. This has no effect on the logs kept in the `/var/opt/SUNWjass/runs` directories. This option is particularly helpful when on a slow terminal connection, as there is a significant amount of output generated by a Toolkit run. This option can be used with either the `-d` or `-u` options.

A jass-execute driver run using the -o option will have output similar to the following:

```
# ./jass-execute -d secure.driver -o jass-output.txt
./jass-execute: NOTICE: Executing driver, secure.driver
./jass-execute: NOTICE: Recording output to jass-output.txt
```

-r root_directory
The -r option can be used to change the root directory used during jass-execute runs. By default, the default root filesystem directory, defined by the Toolkit environment variable JASS_ROOT_DIR, is /. This means that the Solaris OE being secured is available through /. If a separate OS directory, temporarily mounted under /mnt, to be secured, then the -r option can be used to specify
/mnt and all the scripts will be applied to that OS image.

Building Custom jass Packages

As organizations continue to standardize on the Toolkit, a mechanism to easily create a Solaris OE package from only the Toolkit files in a JumpStart server has become increasingly necessary. A script called make-pkg has been included with Toolkit version 0.3 to provide this functionality to enterprise-class Toolkit customers.

The make-pkg script supports the exclusion of top-level directories. Without this functionality it was very cumbersome to create packages at customer environments where the Toolkit was also the JumpStart root, as there would be a fully populated OS, patches, and packages directory in the created package otherwise.

With this feature, administrators can rapidly create packages that exclude directories or files residing in JASS_HOME_DIR without needing to touch the source.

The make-pkg script offers the following options:

-b new-base-dir
Specify an alternate installation base directory

-m new-email-address
Specify an additional e-mail address for support

-t new-title
Specify an additional package title

`-e excl-list`
Exclude top level file or directories from the package. This is done by specifying a '|' separated list as in '`a|b/c|d`'.

`-h`
Display this help message.

The following sample illustrates how a Toolkit package could be created from a JumpStart server:

```
# pwd
/devl
# ./make-pkg -e "OS|Packages|Patches" -b /opt/jass \
-t “Sample jass pkg”
[...]
The following packages are available:
  1  SUNWjass     JASS Toolkit 0.3 / Sample jass pkg
                  (Solaris) 0.3

Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: Transferring <SUNWjass>
package instance

The SUNWjass package has been created as SUNWjass.pkg
```

Based on the alternative `basedir` specified by the `-b` and `-t` options, when run through `pkgadd` the following occurs:

```
# pkgadd -d SUNWjass.pkg

The following packages are available:
  1  SUNWjass     JASS Toolkit 0.3 / Sample jass pkg
                  (Solaris) 0.3

Select package(s) you wish to process (or ’all’ to process
all packages). (default: all) [?,??,q]: 1

Processing package instance <SUNWjass> from </devl/SUNWjass.pkg>

JASS Toolkit 0.3 / Sample jass pkg
(Solaris) 0.3
Using </opt> as the package base directory.
## Processing package information.
## Processing system information.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.

Installing JASS Toolkit 0.3 / Sample jass pkg as <SUNWjass>

## Installing part 1 of 1.
/opt/jass/CHANGES
/opt/jass/CREDITS
/opt/jass/Documentation/BuildInf.pdf
/opt/jass/Documentation/minimize-updt1.pdf
/opt/jass/Documentation/network-updt1.pdf
/opt/jass/Documentation/ntier-security.pdf
/opt/jass/Documentation/security.pdf
[...]
```

Not only does the package install in `/opt/jass` as specified by the `-b` option, but the package name is `JASS Toolkit 0.3 / Sample jass pkg`, as specified by the `-t` option. The argument specified by `-t` is appended to `JASS Toolkit 0.3`.

Conclusion

This article discusses the advanced configuration and usage options available in version 0.3 of the Toolkit. As part of these instructions, the design philosophy of the Toolkit was also reviewed, in addition to discussions of the architecture and framework of the Toolkit.

Specifically, this article describes the configuration options available in the Toolkit. These options have been significantly enhanced in version 0.3. The goal of these options is to minimize the Toolkit code changes required while setting up the Toolkit. New command line options to `jass-execute` added to this latest version of the Toolkit were also described in detail. Each of these options, and particularly the new undo feature, are discussed and sample output provided.

Lastly, the newly added `make-dist` script was described. This feature was added to allow Toolkit users a simple mechanism by which custom-built Toolkit packages can be created for easy distribution within their organizations.

Bibliography

Noordergraaf, Alex and Brunette, Glenn, *The Solaris™ Security Toolkit - Internals: Updated for Toolkit version 0.3,* Sun BluePrints OnLine, June 2001,
`http://www.sun.com/blueprints/0601/jass_internals-v03.pdf`

Noordergraaf, Alex and Brunette, Glenn, *The Solaris™ Security Toolkit - Quick Start: Updated for Toolkit version 0.3,* Sun BluePrints OnLine, June 2001,
`http://www.sun.com/blueprints/0601/jass_quick_start-v03.pdf`

Noordergraaf, Alex and Brunette, Glenn, *The Solaris™ Security Toolkit - Release Notes: Updated for Toolkit version 0.3,* Sun BluePrints OnLine, June 2001,
`http://www.sun.com/blueprints/0601/jass_release_notes-v03.pdf`

Noordergraaf, Alex, *Building a JumpStart™ Infrastructure,* Sun BluePrints OnLine, April 2001,
`http://www.sun.com/blueprints/0401/BuildInf.pdf`

Noordergraaf, Alex, *Solaris™ Operating Environment Minimization for Security: A Simple, Reproducible and Secure Application Installation Methodology - Updated for Solaris 8 Operating Environment*, Sun BluePrints OnLine, November 2000,
`http://www.sun.com/blueprints/1100/minimize-updt1.pdf`

Noordergraaf, Alex and Watson, Keith, *Solaris™ Operating Environment Minimization for Security: A Simple, Reproducible and Secure Application Installation Methodology,* Sun BluePrints OnLine, December 1999,
`http://www.sun.com/blueprints/1299/minimization.pdf`

Noordergraaf, Alex and Watson, Keith, *Solaris™ Operating Environment Network Settings for Security*, Sun BluePrints OnLine, December 2000,
`http://www.sun.com/blueprints/1200/network-updt1.pdf`

Author's Bio: Alex Noordergraaf

Alex Noordergraaf has more than 9 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. Articles completed include recommendations on: Solaris OE Security settings, Solaris OE Minimization, and Solaris OE Network settings.

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 the Sun Professional Services™ organization. His customers have included major telecommunication firms, financial institutions, ISPs, and ASPs.

Author's Bio: Glenn Brunette

Glenn Brunette has more than 8 years experience in the areas of computer and network security. Glenn currently works with in the Sun Professional Services organization where he is a Senior Security Architect and Manager of the North Eastern USA Security Team. In this role, Glenn works with many Fortune 500 companies to deliver tailored security solutions such as assessments, architecture design and implementation, as well as policy and procedure review and development. His customers have included major financial institutions, xSPs, New Media, Life Sciences and government organizations.

In addition to customer delivery services, Glenn works with the Sun Professional Services Global Security Practice and Enterprise Engineering group on the development and review of new security methodologies, best practices, and tools.