Solaris™ Security Toolkit 4.2 Man Page Guide

Sun Microsystems, Inc.
www.sun.com

Part No. 819-1505-10
July 2005, Revision A

Submit comments about this document at: http://www.sun.com/hwdocs/feedback

Please
Recycle

Adobe PostScript

Contents

Preface

The man pages distributed with the Solaris Security 4.2 software are *not* distributed with the Solaris™ Operating System. However, they follow the format of Solaris Operating System man pages. Some Solaris Operating Systems commands are referenced in this guide, and you can find more information about them in the Solaris Reference Manual Collection or man pages. You can use these Solaris Security Toolkit 4.2 man pages to obtain information about the Solaris Security Toolkit and its features.

Overview

The following contains a brief description of each section in the man pages and the information it references:

- Section 1M first lists the command, `Intro`, which you can evoke as a man page in the Solaris Security Toolkit 4.2 software. The `Intro` man page lists the categories of functions and drivers that are supported by Solaris Security Toolkit 4.2 software. Then the section goes on to describe, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes.
- Section 4 outlines the contents of various files.
- Section 7 describes various special files that refer to specific device drivers.

Below is a generic format for man pages. The man pages of each manual section generally follow this order, but include only needed headings. For example, if there is no extended description, there is no EXTENDED DESCRIPTION section. See the `Intro` page for more information and detail about each section, and `man`(1) for more information about man pages in general.

NAME	Provides the names of the commands or functions documented, followed by a brief description of what they do.

SYNOPSIS

Shows the syntax of commands or functions. When a command or file does not exist in the standard path, its full path name is shown. Options and arguments are alphabetized, with single letter arguments first, and options with arguments next, unless a different argument order is required.

The following special characters are used in this section:

`[ ]`	Brackets. The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument must be specified.
…	Ellipses. Several values may be provided for the previous argument, or the previous argument can be specified multiple times, for example "filename...".
|	Separator. Only one of the arguments separated by this character can be specified at one time.
`{ }`	Braces. The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.

DESCRIPTION

Defines briefly what the command does.

EXTENDED DESCRIPTION

Provides more descriptive material. Provides required group privileges, if any.

LIST OF COMMANDS

Lists command, functions, and drivers that are supported.

OPTIONS

Lists the command options with a concise summary of what each option does. The options are listed literally and in the order they appear in the SYNOPSIS section. Possible arguments to options are discussed under the option, and where appropriate, default values are supplied.

EXAMPLES

This section provides examples of usage or of how to use a command or function. Wherever possible a complete example including command line entry and machine response is shown. Whenever an example is given, the prompt is shown as **`example%`** or if the user must be superuser, **`example#`**.

EXIT STATUS

This section lists the values the command returns to the calling program or shell and the conditions that cause these values to be returned. Usually, zero is returned for successful completion and values other than zero for various error conditions.

FILES

This section lists all file names referred to by the man page, files of interest, and files created or required by commands. Each is followed by a descriptive summary or explanation.

ATTRIBUTES

This section lists characteristics of commands, utilities, and device drivers by defining the attribute type and its corresponding value. See `attributes`(5) for more information.

SEE ALSO

This section lists references to other Solaris Security Toolkit man pages.

NAME

Intro - introduce Solaris Security Toolkit administration

SYNOPSIS

Intro

DESCRIPTION

Describes the commands you can execute in the Solaris Security Toolkit, also known as JumpStart Architecture and Security Scripts (JASS).

Sun support for Solaris Security Toolkit software is available only for its use in the Solaris 8, 9, and 10 Operating Systems. While the software can be used in the Solaris 2.5.1, Solaris 2.6, and Solaris 7 Operating Systems, Sun support is not available for its use in those operating systems.

The Solaris Security Toolkit software automatically detects which version of the Solaris Operating System software is installed, then runs tasks appropriate for that operating system version.

LIST OF COMMANDS

The following commands, functions, and drivers are supported by the Solaris Security Toolkit 4.2 software:

`Intro` Lists Solaris Security Toolkit commands, functions, and drivers.

`add-client` Simplifies adding JumpStart™ clients to a JumpStart server that has Solaris Security Toolkit installed. `add-client` is a wrapper around the `add_install_client` script.

`audit_public_funcs` Lists all public audit functions for the Solaris Security Toolkit that are in the `audit_public.funcs` file.

`common_log_funcs` Lists all common log functions in `common_log.funcs` file that control all logging and reporting Solaris Security Toolkit functions.

`common_misc_funcs` Lists all miscellaneous framework Solaris Security Toolkit functions in `common_misc.funcs` file.

`driver_public_funcs` Lists all public functions for the Solaris Security Toolkit drivers that are in the `driver_public.funcs` file.

`jass-check-sum` Identifies file changes made since the last Security Toolkit hardening run by using checksums.

`jass-execute` Performs most of the functionality of the Solaris Security Toolkit software.

`make-jass-pkg`
Allows the creation of a customized Solaris Security Toolkit package from a customized version installed on a system.

`rm-client`
Simplifies removing JumpStart clients from a JumpStart server that has Solaris Security Toolkit installed. `rm-client` is a wrapper around the `rm_install_client` script.

`security_drivers`
Lists all Solaris Security Toolkit drivers in the `security.drivers` file in the `Drivers` directory.

NAME

add-client - install JumpStart client for the Solaris Security Toolkit

SYNOPSIS

add-client `-c` *client-host-name* [`-i` *install-server*] [`-m` *client-mach-class*] [`-o` *solaris-os-instance*] [`-s` *sysidcfg-dir*]

add-client `-?|-h`

add-client `-v`

DESCRIPTION

`add-client` is a wrapper around the `add_install_client` script, which simplifies adding JumpStart clients to a JumpStart server that has Solaris Security Toolkit installed. The command is located in the `bin` directory of the Solaris Security Toolkit distribution package.

EXTENDED DESCRIPTION

For SPARC-based systems, the `add-client` command installs the JumpStart client and configuration information needed by the Solaris Security Toolkit. The command is executed from the JumpStart server.

For x86 systems, which use Dynamic Host Configuration Protocol (DHCP) clients, you need to use the `add_install_client` script provided with the Solaris (Install) Media. This also applies to JumpStart configurations that need to use advanced JumpStart features not included in the `add-client` script, such as performing the necessary JumpStart configuration for clients.

Group Privileges Required

You must have superuser privileges to run this command.

OPTIONS

The following options are supported:

`-c` *client-host-name*
Specifies the name of the JumpStart client to be installed.

`-h |-?`
Displays usage descriptions.

Use alone. Any option specified in addition to `-h` or `-?` is ignored.

`-i` *install-server*
Specifies the name of the JumpStart install server. If no value is given, a list of available options is provided. If the system has only one network interface then `add-client` uses it by default.

`-m` *client-mach-class*
Specifies the machine class of the JumpStart client. This value must be in the same format as the output of the `uname -n` command. If not specified, the default of `sun4u` is used.

`-o` *solaris-os-instance*
Specifies the version of the Solaris Operating System to be installed on the client. If no value is given, a list of available options is provided. If only one instance is available, `add-client` uses it by default.

`-s` *sysidcfg-dir* Specifies the path name to an alternate directory in which a system identification and configuration (`sysidcfg`) file is stored. By default, the value is set to the directory, `JASS_HOME/Sysidcfg/`*Solaris-ver*`/`. If this option is used, this path name should be specified relative to the `JASS_HOME/Sysidcfg` directory. For example, `Hosts/alpha` where `JASS_HOME/Sysidcfg/Hosts/alpha` exists and contains a `sysidcfg` file.

`-v` Displays the version information for this program.

EXAMPLES

EXAMPLE 1 Adding a Client to a System Using Defaults

```
sc0:#:> /opt/SUNWjass/bin/add-client -c eng1 -m sun4u
Selecting default operating system, Solaris_ver.
Selecting default system interface, IP_address.
cleaning up preexisting install client "eng1"
removing eng1 from bootparams
updating /etc/bootparams
sc0:#:>
```

where:

Solaris_ver Only version of the Solaris OS installed in `JASS_HOME_DIR/OS`.

IP_address Only network interface of the system on which the command was run. Written as four sets of numbers separated by periods; for example, 172.16.0.59.

`eng1` Host name of the JumpStart client.

EXAMPLE 2 Add a Client to a System Using Full Options

```
sc0:#:> /opt/SUNWjass/bin/add-client -c eng1 -i jumpserve1 -m
sun4u -o Solaris_9_2003-12 -s Hosts/alpha
cleaning up preexisting install client "eng1"
removing eng1 from bootparams
updating /etc/bootparams
sc0:#:>
```

where:

`eng1` Host name of the JumpStart client.

`jumpserve1` Name of the local interface on sc0, through which the JumpStart client is installed.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
1	Error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes.

Attribute Types	Attribute Values
Availability	SUNWjass
Interface Stability	Unstable

SEE ALSO

jass-check-sum(1M)

jass-execute(1M)

make-jass-pkg(1M)

rm-client(1M)

NAME

audit_public_funcs - list all public audit functions in `audit_public.funcs` file

SYNOPSIS

`audit_public_funcs`

DESCRIPTION

All auditing functions used in audit scripts are located in the `Drivers` directory in the `audit_public.funcs` file. Functions defined in this file are public and can be freely used in both standard and custom audit scripts. In many cases, the functions defined in this file are stubs that call functions defined in the `audit_private.funcs` file. These stubs were implemented to allow users to code their scripts to these public interfaces without needing to know if the underlying code is modified or enhanced in later releases.

Framework functions provide flexibility for you to change the behavior of the Solaris Security Toolkit software without modifying source code.

EXTENDED DESCRIPTION

Note – Two types of audit functions are in the software: private and public. The functions defined in the `audit_private.funcs` file are private and *not* for public use. *Never* use the private scripts defined in this file. Only use the public scripts defined in the `audit_public.funcs` file.

Use these functions as part of audit scripts to assess components of the system's stored and run-time configurations. These functions are public interfaces to the Solaris Security Toolkit software's audit framework.

When customizing or creating audit scripts, use the following functions to perform standard operations:

- `check_fileContentsExist` and `check_fileContentsNotExist`
- `check_fileExists` and `check_fileNotExists`
- `check_fileGroupMatch` and `check_fileGroupNoMatch`
- `check_fileModeMatch` and `check_fileModeNoMatch`
- `check_fileOwnerMatch` and `check_fileOwnerNoMatch`
- `check_fileTemplate`
- `check_fileTypeMatch` and `check_fileTypeNoMatch`
- `check_if_crontab_entry_present`
- `check_keyword_value_pair`
- `check_minimized`
- `check_minimized_service`
- `check_packageExists` and `check_packageNotExists`
- `check_patchExists` and `check_patchNotExists`
- `check_processArgsMatch` and `check_processArgsNoMatch`
- `check_processExists` and `check_processNotExists`
- `check_serviceConfigExists` and `check_serviceConfigNotExists`

- check_serviceDisabled and check_serviceEnabled
- check_serviceInstalled and check_serviceNotInstalled
- check_serviceOptionDisabled and check_serviceOptionEnabled
- check_servicePropDisabled
- check_serviceRunning and check_serviceNotRunning
- check_startScriptExists and check_startScriptNotExists
- check_stopScriptExists and check_stopScriptNotExists
- check_userLocked and check_userNotLocked
- finish_audit
- get_cmdFromService
- start_audit

For detailed information and instructions on the use of each of these functions please refer to the "Framework Functions" chapter of the *Solaris Security Toolkit 4.2 Reference Manual*.

EXAMPLES

EXAMPLE 1 Checking for the Existence of a File

```
check_fileExists /etc/inet/inetd.conf 1 LOG
```

EXAMPLE 2 Checking for the Existence of a Package

```
check_packageExists SUNWsshdu 1 LOG
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes.

ATTRIBUTE TYPE	ATTRIBUTE VALUES
Availability	SUNWjass
Stability	Unstable

SEE ALSO

add-client(1M)

common_log_funcs(4)

common_misc_funcs(4)

driver_public_funcs(4)

jass-check-sum(1M)

jass-execute(1M)

make-jass-pkg(1M)

rm-client(1M)

security_drivers(7)

NAME

common_log_funcs - list all common log functions in the `common_log.funcs` file

SYNOPSIS

`common_log_funcs`

DESCRIPTION

All logging and reporting functions are located in the `Drivers` directory in a file called `common_log.funcs`. The logging and reporting functions are used in all of the Solaris Security Toolkit software's operational modes; therefore, they are considered common functions. For example, functions such as `logWarning` and `logError` are in this file.

Framework functions provide flexibility for you to change the behavior of the Solaris Security Toolkit software without modifying source code.

EXTENDED DESCRIPTION

The following is a list of common log functions:

- `logBanner`
- `logDebug`
- `logError`
- `logFailure`
- `logFileContentsExist` and `logFileContentsNotExist`
- `logFileExists` and `logFileNotExists`
- `logFileGroupMatch` and `logFileGroupNoMatch`
- `logFileModeMatch` and `logFileModeNoMatch`
- `logFileNotFound`
- `logFileOwnerMatch` and `logFileOwnerNoMatch`
- `logFileTypeMatch` and `logFileTypeNoMatch`
- `logFinding`
- `logFormattedMessage`
- `logInvalidDisableMode`
- `logInvalidOSRevision`
- `logMessage`
- `logNotGlobalZone`
- `logNotice`
- `logPackageExists` and `logPackageNotExists`
- `logPatchExists` and `logPatchNotExists`
- `logProcessArgsMatch` and `logProcessArgsNoMatch`
- `logProcessExists` and `logProcessNotExists`
- `logProcessNotFound`
- `logScore`

- `logScriptFailure`
- `logServiceConfigExists` and `logServiceConfigNotExists`
- `logServiceDisabled` and `logServiceEnabled`
- `logServiceInstalled` and `logServiceNotInstalled`
- `logServiceOptionDisabled` and `logServiceOptionEnabled`
- `logServiceProcessList`
- `logServicePropDisabled` and `logServicePropEnabled`
- `logServiceRunning` and `logServiceNotRunning`
- `logStartScriptExists` and `logStartScriptNotExists`
- `logStopScriptExists` and `logStopScriptNotExists`
- `logSuccess`
- `logSummary`
- `logUndoBackupWarning`
- `logUserLocked` and `logUserNotLocked`
- `logWarning`

For detailed information and instructions on the use of each of these functions please refer to the "Framework Functions" chapter of the *Solaris Security Toolkit 4.2 Reference Manual*.

EXAMPLES

EXAMPLE 1 Logging a Log Failure

```
Usage:
logFailure "Package SUNWatfsr is installed."
Output:
[FAIL] Package SUNWatfsr is installed.
```

EXAMPLE 2 Logging a Log File Existence

```
Usage:
logFileExists /etc/issue
Output:
[NOTE] File /etc/issue was found.
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes.

ATTRIBUTE TYPE	ATTRIBUTE VALUES
Availability	SUNWjass
Stability	Unstable

SEE ALSO

add-client(1M)

audit_public_funcs(4)

common_misc_funcs(4)

driver_public_funcs(4)

jass-check-sum(1M)

jass-execute(1M)

make-jass-pkg(1M)

rm-client(1M)

security_drivers(7)

NAME

common_misc_funcs - list miscellaneous framework functions in the `common_misc.funcs` file

SYNOPSIS

`common_misc_funcs`

DESCRIPTION

Miscellaneous functions are used within several areas of the Solaris Security Toolkit software and are not specific to functionality provided by other framework functions. The miscellaneous functions are in the `Drivers` directory in a file called `common_misc.funcs`. Common utility functions such as `isNumeric` and `printPretty` are in this file.

Framework functions provide flexibility for you to change the behavior of the Solaris Security Toolkit software without modifying source code.

EXTENDED DESCRIPTION

The following is a list of common miscellaneous functions:

- `adjustScore`
- `checkLogStatus`
- `clean_path`
- `extractComments`
- `get_driver_report`
- `get_lists_conjunction`
- `get_lists_disjunction`
- `invalidVulnVal`
- `isNumeric`
- `printPretty`
- `printPrettyPath`
- `strip_path`

For detailed information and instructions on the use of each of these functions please refer to the “Framework Functions” chapter of the *Solaris Security Toolkit 4.2 Reference Manual*.

ATTRIBUTES

See **attributes**`(5)` for descriptions of the following attributes.

ATTRIBUTE TYPE	ATTRIBUTE VALUES
Availability	SUNWjass
Stability	Unstable

SEE ALSO

add-client`(1M)`

audit_public_funcs`(1M)`

common_log_funcs(4)

driver_public_funcs(4)

jass-check-sum(1M)

jass-execute(1M)

make-jass-pkg(1M)

rm-client(1M)

security_drivers(7)

NAME

driver_public_funcs - lists driver functions found in the driver_public.funcs file

SYNOPSIS

driver_public_funcs

DESCRIPTION

All functions that control Solaris Security Toolkit driver functionality are located in the Drivers directory in the driver_public.funcs file. Functions such as add_pkg and copy_a_file are in this file.

EXTENDED DESCRIPTION

When customizing or creating scripts, use the following functions to perform standard operations:

- add_crontab_entry_if_missing
- add_option_to_ftpd_property
- add_patch
- add_pkg
- add_to_manifest
- backup_file
- backup_file_in_safe_directory
- change_group
- change_mode
- change_owner
- check_and_log_change_needed
- check_os_min_version
- check_os_revision
- check_readOnlyMounted
- checksum
- convert_inetd_service_to_fmri
- copy_a_dir
- copy_a_file
- copy_a_symlink
- copy_files
- create_a_file
- create_file_timestamp
- disable_conf_file
- disable_file
- disable_rc_file
- disable_service

- enable_service
- find_sst_run_with
- get_expanded_file_name
- get_stored_keyword_val
- get_users_with_retries_set
- is_patch_applied and is_patch_not_applied
- is_service_enabled
- is_service_installed
- is_service_running
- is_user_account_extant
- is_user_account_locked
- is_user_account_login_not_set
- is_user_account_passworded
- lock_user_account
- make_link
- mkdir_dashp
- move_a_file
- rm_pkg
- set_service_property_value
- set_stored_keyword_val
- unlock_user_account
- update_inetcon_in_upgrade
- warn_on_default_files
- write_val_to_file

For detailed information and instructions on the use of each of these functions please refer to Chapter 2, “Framework Functions”, of the *Solaris Security Toolkit 4.2 Reference Manual*.

EXAMPLES

EXAMPLE 1 Adding a Single Patch

```
add_patch 123456-01
```

EXAMPLE 2 Adding a Patch List

```
add_patch -M ${JASS_PATCH_DIR}/OtherPatches patch_list.txt
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes.

ATTRIBUTE TYPE	ATTRIBUTE VALUES
Availability	SUNWjass

SEE ALSO

add-client(1M)

audit_public_funcs(4)

common_log_funcs(4)

common_misc_funcs(4)

jass-check-sum(1M)

jass-execute(1M)

make-jass-pkg(1M)

rm-client(1M)

security_drivers(7)

NAME

jass-check-sum - identify file changes made since the last Solaris Security Toolkit hardening run

SYNOPSIS

jass-check-sum

DESCRIPTION

This Solaris Security Toolkit script identifies those files that have been modified since their checksums were last saved in the `JASS_REPOSITORY` (`/var/opt/SUNWjass/run/*/jass-checksums.txt`).

Only the most recent checksum of a file is compared to the current file. This aids in determining if a file has been changed after being configured by the Solaris Security Toolkit. If a given configuration has already been undone, this script skips it.

EXTENDED DESCRIPTION

Group Privileges Required

You should have superuser privileges to run this command.

OPTIONS

None.

EXAMPLES

EXAMPLE 1 Checking the Solaris Security Toolkit Files

```
sc0: #:> /opt/SUNWjass/bin/jass-check-sum

Checking for file signature conflicts associated with Toolkit run:
20040621172054

File Name                 Saved CkSum                Current CkSum
-----------------------------------------------------------------------
/etc/passwd                685593234:456                1703916610:489
/etc/shadow                3216256103:185               3154547236:190

sc0:#:>
```

EXIT STATUS

The following exit values are returned:

0	Successful completion.
1	Error occurred.

FILES

The following `JASS_REPOSITORY` file is used by this command.

`/var/opt/SUNWjass/run/`*run-id*`/jass-checksums.txt`	Contains list of files which are compared to files being tested.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes.

Attribute Types	Attribute Values
Availability	SUNWjass
Interface Stability	Evolving

SEE ALSO

add-client(1M)

audit_public_funcs (4)

common_log_funcs(4)

common_misc_funcs(4)

driver_public_funcs(4)

jass-execute(1M)

make-jass-pkg(1M)

rm-client(1M)

NAME

jass-execute - execute the Solaris Security Toolkit functionality

SYNOPSIS

jass-execute [`-r` *root_directory* `-p` *os_version*] [`-q`|`-o` *output_file*] [`-m` *e-mail_address*] [`-V` [3|4]] [`-d` *driver*]

jass-execute `-u` [`-b`|`-f`|`-k`] [`-q`|`-o` *output_file*] [`-m` *e-mail_address*] [`-V [3|4]`]

jass-execute `-a` *driver* [`-V` [0|1|2|3|4]] [`-q`|`-o` *output_file*] [`-m` *e-mail_address*]

jass-execute `-c` [`-q`|`-o` *output_file*] [`-m` *e-mail_address*] [`-V [3|4]`]

jass-execute `-H`

jass-execute `-l`

jass-execute `-h|-?`

jass-execute `-v`

DESCRIPTION

`jass-execute` executes various functions of the Solaris Security Toolkit (also known as JASS) depending on the options used. For more information about how to use the `jass-execute` command and its various options, refer to the *Solaris Security Toolkit 4.2 Administration Guide,* Chapter 3, “Upgrading, Installing, and Running Security Software.”

EXTENDED DESCRIPTION

You need to specify a driver with options `-a` and `-d` of the `jass-execute` command. Drivers are used by the Solaris Security Toolkit software to harden, minimize, and audit Solaris OS systems. A series of drivers and related files make up a security profile.

The following standard drivers are supplied by default in the `Drivers` directory:

- `[secure|hardening|config].driver`

The following product-specific drivers are used to harden specific Sun products or configurations.

- `server-[secure|hardening|config].driver`
- `suncluster3x-[secure|hardening|config].driver`
- `sunfire_15k_sc-[secure|hardening|config].driver`

Note – Use *only* the [*name*]-secure.driver as an argument to the `jass-execute` command.

For more information about drivers, refer to the *Solaris Security Toolkit 4.2 Reference Manual*, Chapter 4, “Drivers”.

Group Privileges Required

You must have superuser privileges to run this command.

OPTIONS

The following options are supported:

`-a` *driver*	Determines if the system is in compliance with its security profile. Do *not* use with the `-c`, `-d`, `-h`, `-H`, `-l`, `-p`, `-r`, or `-u` options.
`-b`	Backs up any files that have been manually changed since the last hardening run, then restores system to its original state. Use *only* with the `-u` option.
`-c`	Specifies the clean option. Removes saved files from a previous run of Solaris Security Toolkit.
`-d` *driver*	Specifies the driver to be run in stand-alone mode. Do *not* use with the `-a`, `-b`, `-c`, `-f`, `-h`, `-H`, `-k`, or `-u` options.
`-f`	Reverses changes made during a hardening run without asking you about exceptions, even if files were manually changed after a hardening run. Use *only* with the `-u` option.
`-H`	Displays the history of Solaris Security Toolkit applications on the system.
`-h |-?`	Displays usage descriptions for `jass-execute`. Use alone. Any option specified in addition to `-h|-?` is ignored.
`-k`	Keeps any manual changes you made to files after a hardening run. Use *only* with the `-u` option.
`-l`	Displays the last application of the Solaris Security Toolkit installed on the system.
`-m` *e-mail_address*	Specifies an email address for in-house support.
`-o` *output_file*	Specifies the complete path to the output file as well as the output file itself.
`-p` *os_version*	Specifies the OS version of Solaris. The format is the same as that of `uname -r`. *Must* use with the `-r` *root_directory* option.

`-q`

Specifies quiet mode. Messages are not displayed while running this command. Output is stored in `JASS_REPOSITORY/`.

`-r` *root_directory*

Specifies the root directory used during jass-execute runs. By default, the root file system is `/`. This root directory is defined by the Solaris Security Toolkit environment variable, `JASS_ROOT_DIR`. The Solaris OS being secured is available through `/`. For example, if you wanted to secure a separate OS directory, temporarily mounted under `/mnt` then use the `-r` option to specify `/mnt`.

Must use with the `-p` *os_version* option.

`-u`

Runs the undo option with interactive prompts that ask you what action you want to take when exceptions are encountered.

Do *not* use with the `-a`, `-c`, `-d`, `-h`, `-l`, or `-H` options.

`-V` *verbosity_level*

Specifies the level of verbosity for an audit run. There are five levels (0-4):

`0` Final. This mode results in only one line of output that indicates the combined result of the entire verification run. This mode is useful if a single PASS or FAIL is needed.

`1` Consolidated. In this mode, one line of output per audit script is generated indicating the result of each audit script. In addition, subtotals are generated at the end of each script, as well as a grand total at the end of the run.

`2` Brief. This mode combines the attributes of the Consolidated verbosity level and includes the results of the individual checks within each audit script. This mode is useful for quickly determining those items that passed and failed within a single audit script. The format of this mode still represents one result per line.

`3` Full. This is the first of the multiline verbosity modes. In this mode, banners and headers are printed to illustrate more clearly the checks that are being run, their intended purpose, and how their results are determined. This is the default verbosity level and more suitable for those new to the Solaris Security Toolkit verification capability.

`4` Debug. This mode extends upon the Full verbosity mode by including all entries that are generated by the logDebug logging function. Currently, this is not used by any of the Solaris Security Toolkit audit scripts, but it is included for completeness and to allow administrators to embed debugging statements within their code.

`-v`

Displays the version information for this program.

EXAMPLES

EXAMPLE 1 Configuring a Solaris Security Toolkit Application

```
sc0:#:> /opt/SUNWjass/bin/jass-execute -r /mnt -p 5.9 -o
output.txt -m support@mycompany.com -d secure.driver

[NOTE] The following prompt can be disabled by setting JASS_NOVICE_USER
to 0.
[WARN] Depending on how the Solaris Security Toolkit is configured, it is
both possible and likely that by default all remote shell and file transfer
access to this system will be disabled upon reboot effectively locking out
any user without console access to the system.
Are you sure that you want to continue? (YES/NO) [NO] YES
[NOTE] Executing driver, secure.driver
[NOTE] Recording output to output.txt
sc0:#:>
```

EXAMPLE 2 Undoing a Previous Jass Application

```
sc0:#:> /opt/SUNWjass/bin/jass-execute -u -b -q -m
support@mycompany.com -V 3
[WARN] Creating backup copies of some files may cause unintended affects.
[WARN] This is particularly true of /etc/hostname.[interface] files as
well as crontab files in /var/spool/cron/crontabs.
[NOTE] Executing driver, undo.driver

Please select a Solaris Security Toolkit run to restore through:
1.  June 28, 2004 at 19:11:49 (/var/opt/SUNWjass/run/20040628191149)
2.  June 21, 2004 at 17:20:54 (/var/opt/SUNWjass/run/20040621172054)
3.  June 17, 2004 at 10:45:23 (/var/opt/SUNWjass/run/20040617104523)
Choice ('q' to exit)? 1
[NOTE] Restoring to previous run from
/var/opt/SUNWjass/run/20040628191149
sc0:#:>
```

EXAMPLE 3 Auditing the System Against a Pre-Defined Profile

```
sc0:#:> /opt/SUNWjass/bin/jass-execute -a secure.driver -V 2 -o
output.txt -m support@mycompany.com

jass-execute                   [NOTE] Executing driver, secure.driver
jass-execute                   [NOTE] Recording output to output.txt
sc0:#:>
```

EXAMPLE 4 Displaying the Last Installed Solaris Security Toolkit Application

```
sc0:#:> /opt/SUNWjass/bin/jass-execute -l

# ./jass-execute -l
This information is only applicable for applications of the
Solaris Security Toolkit starting with version 0.3.
The last application of the Solaris Security Toolkit was:
1.   June 28, 2004 at 19:11:49 (20040628191149) (UNDONE)
sc0:#:>
```

EXIT STATUS

The following exit values are returned:

0	Successful completion.
1	Error occurred.
2	Security violation occurred.
3	Another instance of jass-execute is running.
4	Termination by user request.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes.

Attribute Types	Attribute Values
Availability	SUNWjass
Interface Stability	Evolving

SEE ALSO

add-client(1M)

jass-check-sum(1M)

make-jass-pkg(1M)

rm-client(1M)

NAME

make-jass-pkg - create Solaris Security Toolkit (JASS) package stream file

SYNOPSIS

make-jass-pkg [-b *new-base-dir*] [-e *excl-list*] [-m *new-email-address*] [-p *package-name*] [-q] [-t *new-title*]

make-jass-pkg -v

make-jass-pkg -?|-h

DESCRIPTION

The make-jass-pkg command creates a Solaris package stream file from the Solaris Security Toolkit distribution. The resulting file can be installed using the pkgadd command and removed using the pkgrm command. Information about the installed distribution can be obtained using the pkginfo command.

EXTENDED DESCRIPTION

Group Privileges Required

You must have superuser privileges to run this command.

OPTIONS

The following options are supported:

-b *new-base-dir*
Specifies an alternate installation base directory.

-e *excl-list*
Excludes top level files or directories from the package. This is done by specifying a pipe (|) separated list; for example, a|b|c|d.

-h |-?
Displays usage descriptions.

Use alone. Any option specified in addition to -h or -? is ignored.

-m *new-email-address*
Specifies an email address to use for in-house support.

-p *package-name*
Specifies a custom package name. The default is JASScustm.

-q
Specifies quiet mode. No messages are displayed when this command is run.

-t *new-title*
Specifies an alternative package title. The default title is "Solaris Security Toolkit".

-v
Displays the version information for this program.

EXAMPLES

EXAMPLE 1 Creating a Package Stream File Using Defaults

```
sc0: #:> /opt/SUNWjass/bin/make-jass-pkg

[NOTE] Creating the package’s prototype file.  This may take a few minutes.
[NOTE] Excluded file: ./jass-include-list.tmp
[NOTE] Creating the package’s info file.
[NOTE] Creating the package in a scratch directory.
## Building pkgmap from package prototype file.
## Processing pkginfo file.
WARNING: parameter <PSTAMP> set to "eng120040623143146"
WARNING: parameter <CLASSES> set to "none"
## Attempting to volumize 360 entries in pkgmap.
part  1 -- 2934 blocks, 357 entries
## Packaging one part.
/opt/SUNWjass/SUNWjass/pkgmap
/opt/SUNWjass/SUNWjass/pkginfo
.
.[list of files...]
.
/opt/SUNWjass/SUNWjass/reloc/rules.SAMPLE
/opt/SUNWjass/SUNWjass/install/tsolinfo
## Validating control scripts.
## Packaging complete.
[NOTE] Creating the package’s stream format (package file).
The following packages are available:
  1 JASScustm Solaris Security Toolkit 4.1.0
                  (Solaris) 4.1.0
Select package(s) you wish to process (or ’all’ to process
all packages). (default: all) [?,??,q]: Transferring <JASScustm> package
instance
[NOTE] The package has been created as JASScustm.pkg.
sc0: #:>
```

EXAMPLE 2 Creating a Package Stream File and Specifying Options

```
sc0: #:> /opt/SUNWjass/bin/make-jass-pkg -b /opt/SUNWjass/otherdir -e
/opt/SUNWjass/test -m eng_support@mycompany.com -p MYJASS -t MyToolkit

[NOTE] Creating the package's prototype file.  This may take a few
minutes.
[NOTE] Creating the package's info file.
[NOTE] Creating the package in a scratch directory.
## Building pkgmap from package prototype file.
## Processing pkginfo file.
WARNING: parameter <PSTAMP> set to "eng120040623150621"
WARNING: parameter <CLASSES> set to "none"
## Attempting to volumize 363 entries in pkgmap.
part  1 -- 5612 blocks, 359 entries
## Packaging one part.
/opt/SUNWjass/SUNWjass/pkgmap
/opt/SUNWjass/SUNWjass/pkginfo
.
.
.[list of files]
/opt/SUNWjass/SUNWjass/reloc/rules.SAMPLE
/opt/SUNWjass/SUNWjass/install/tsolinfo
## Validating control scripts.
## Packaging complete.
[NOTE] Creating the package's stream format (package file).
The following packages are available:
  1 MYJASS Solaris Security Toolkit 4.1.0 / MyToolkit
                  (Solaris) 4.1.0
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: Transferring <MYJASS> package
instance
[NOTE] The package has been created as MYJASS.pkg.
sc0: #:>
```

EXIT STATUS

The following exit values are returned:

0	Successful completion.
1	Error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes.

Attribute Types	Attribute Values
Availability	SUNWjass
Interface Stability	Evolving

SEE ALSO

add-client(1M)

jass-check-sum(1M)

jass-execute(1M)

rm-client(1M)

NAME

rm-client - remove JumpStart client for the Solaris Security Toolkit

SYNOPSIS

rm-client [-c] *client-host-name*

rm-client -? | -h

rm-client -v

DESCRIPTION

rm-client simplifies removing JumpStart clients from a JumpStart server that has Solaris Security Toolkit installed. The rm-client command is a wrapper around the rm_install_client script, and is located in the bin directory of the Solaris Security Toolkit distribution package.

EXTENDED DESCRIPTION

Group Privileges Required

You must have superuser privileges to run this command.

OPTIONS

The following options are supported:

-c *client-host-name*	Removes the installed JumpStart client as well as all configuration information with it, needed by the Solaris Security Toolkit.
-h | -?	Displays usage descriptions. Use alone. Any option specified in addition to -h or -? is ignored.
-v	Displays the version information for this program.

EXAMPLES

EXAMPLE 1 Removing Client

```
sc0: #:> /opt/SUNWjass/bin/rm-client -c eng1
removing eng1 from bootparams
```

where:

eng1	Host name of the client to be removed.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
1	Error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes.

Attribute Types	Attribute Values
Availability	SUNWjass
Interface Stability	Unstable

SEE ALSO

add-client(1M)

jass-check-sum(1M)

jass-execute(1M)

make-jass-pkg(1M)

NAME

security_drivers - list the standard Solaris Security Toolkit drivers found in the `security.drivers` file

SYNOPSIS

`secure.driver`

`server-secure.driver`

`suncluster3x-secure.driver`

`sunfire_15k_sc-secure.driver`

DESCRIPTION

`security_drivers` lists the collection of drivers used by the Solaris Security Toolkit found in the `security.drivers` file.

EXTENDED DESCRIPTION

The following list describes briefly the standard drivers:

- `secure.driver` is the default driver used in the rules for client installation. Implements all the hardening functionality.
- `server-secure.driver` is based on the secure.driver, and highlights what might be necessary to secure server systems.
- `suncluster3x-secure.driver` provides a baseline configuration for hardening Sun™ Cluster 3.x software releases.
- `sunfire_15k_sc-secure.driver` is the only supported mechanism by which the Sun Fire high-end system controller can be secured.

For detailed information and instructions on the use of each of these drivers please refer to the Chapter 4, “Drivers”, in the *Solaris Security Toolkit 4.2 Reference Manual*.

EXAMPLES

EXAMPLE 1 Contents of the `secure.driver` File

```
DIR="`/bin/dirname $0`"
export DIR

. ${DIR}/driver.init

. ${DIR}/config.driver

. ${DIR}/hardening.driver
```

ATTRIBUTES

See **attributes**`(5)` for descriptions of the following attributes.

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWjass
Stability	Unstable

SEE ALSO

add-client`(1M)`

audit_public_funcs(4)

common_log_funcs(4)

common_misc_funcs(4)

driver_public_funcs(4)

jass-check-sum(1M)

jass-execute(1M)

make-jass-pkg(1M)

rm-client(1M)