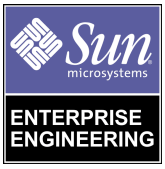

# JumpStart™ Architecture and Security Scripts for the Solaris™ Operating Environment - Part 2

*Updated for Toolkit version 0.2*

---

*By Alex Noordergraaf - Enterprise Engineer and Glenn Brunette - SunPS*

*Sun BluePrints™ OnLine - November 2000*




**http://www.sun.com/blueprints**

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

Part No.: 806-6475-10
Revision 02, November 2000

Please
Recycle

# JumpStart™ Architecture and Security Scripts for the Solaris™ Operating Environment - Part 2

*Updated for Toolkit version 0.2*

## Update

This Sun BluePrints™ OnLine article has been updated to reflect changes in the newly released version (0.2) of the JumpStart™ Architecture and Security Scripts (“JASS” Toolkit) for the Solaris™ Operating Environment.

Parts 1, 2, and 3 of the *JumpStart™ Architecture and Security Scripts for the Solaris™ Operating Environment - Updated for version 0.2* articles are available at:

- *Part 1* – `http://www.sun.com/blueprints/1100/jssec-updt1.pdf`
- *Part 2* – `http://www.sun.com/blueprints/1100/jssec2-updt1.pdf`
- *Part 3* – `http://www.sun.com/blueprints/1100/jssec3-updt1.pdf`

## Overview

This is the second article in a three part series discussing the JumpStart Architecture and Security Scripts (“JASS“ Toolkit). The Toolkit is a recommended mechanism of securing systems using the Solaris Operating Environment (Solaris OE).

The first article presented a detailed overview of the JumpStart product, and provided step-by-step instructions for installing and configuring a JumpStart client and server.

This article continues with an overview of the configuration files, directories, and scripts used by the Toolkit to harden Solaris systems.

# 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 which will reduce the number of components that have to be patched and made secure—reducing the number of components has the potential to reduce entry points to an intruder.

---

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

---

The Toolkit was designed to harden systems during installation—this is achieved by using the JumpStart technology as a mechanism for running the Toolkit scripts. As discussed in Part 1 of this series, JumpStart technology provides a method of installing the Solaris OE over a network, and also has the ability to run scripts on the JumpStart client during installation. This function is used to run the scripts in the Toolkit during the installation process automatically.

Additionally, the Toolkit has been designed to run outside the JumpStart framework in a standalone mode. This standalone mode allows the Toolkit to be used on systems that require security modifications and/or updates but cannot be taken out of service to re-install the OS from scratch.

The Toolkit was built with a modular framework. Customers with existing JumpStart installations will benefit from the Toolkit's ability to integrate into their existing JumpStart architecture. For customers who do not currently use the JumpStart product, the flexibility of the Toolkit's framework will enhance their ability to start using the JumpStart product efficiently.

# 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.

# Toolkit Architecture

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

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

## `Documentation` Directory

This directory contains all Sun BluePrints Online documentation discussing security recommendations for the Toolkit. It may also be accessed through the Internet from:

```
http://www.sun.com/blueprints/browsesubject.html#security
```

## `Drivers` Directory

This directory contains all driver scripts—driver scripts are the scripts listed in the `rules` files that call all other scripts during Toolkit execution. The driver scripts determine which security modifications will be made to each system by calling specified finish scripts. The finish scripts perform the actual modifications to the Solaris OE on the JumpStart clients.

## `Files` Directory

This directory stores files to be copied to the JumpStart client—the `Files` directory is used in conjunction with an environment variable and driver scripts to select and copy files to the JumpStart client.

## `Finish` Directory

This directory contains the finish scripts that perform system modifications and updates during installation. Finish scripts have been written to perform various tasks such as patch and software installation. These scripts will be discussed further in Part 3 of this series.

## `OS` Directory

This directory must contain only Solaris OE files. These files will be used by the JumpStart server (over the network) to build the client. Different Solaris OE releases should be stored in sub directories within this subdirectory. The sub directories should use the naming convention recommended in Part 1 to enable fine grained control for testing and deployment purposes.

## `Packages` Directory

This directory contains software packages that will be installed by the finish scripts. For example, Secure Shell (SSH) software could be stored in the `Packages` directory so the appropriate finish script can install and configure the software as required.

## `Patches` Directory

This directory contains the Recommended and Security Patch Clusters (in addition to individual patches). Sub directories should be created in the `Patches` directory for each of the Solaris OE versions being used. The patch clusters should be extracted into the individual sub directories—this will allow the patch installation script to run without having to first extract the patch cluster for each system installation.

## `Profiles` Directory

This directory contains all profiles—a profile is a file that contains configuration information used by the JumpStart software to determine which Solaris OE cluster to install (`Core`, `End User`, `Developer`, or `Entire Distribution`), the disk layout to use, and the type of installation to perform (e.g., standalone). These configuration files are used to define how specific systems, or groups of systems are built. Part 1 of this series discusses `Profiles` in greater detail.

## `Sysidcfg` Directory

This directory contains directories with OS and host specific `sysidcfg` files. Due to the OS specific nature of the sysidcfg file, a generic version can no longer be used for all Solaris OE releases. The sub directories should use a naming convention similar to that recommended for the `OS` directory in Part 1. The installation convention used is `Solaris_x.x<version #>`. The `sysidcfg` files for the Solaris OE version 2.6 should be stored in a sub-directory named `Solaris_2.6`.

# Script Development Framework

The JumpStart software determines which Solaris OE cluster type to install, specifies disk partitioning, and calls all scripts that are to be executed based on the information specified in the `rules` file. Additionally, it provides a robust framework for developing scripts to configure Solaris OE systems.

The Toolkit architecture includes additional configuration information that enables scripts to be used in different environments. All variables used in the scripts are maintained in a configuration file—this configuration file is imported by a driver script which will then make the variables available to all subsequent scripts.

# Toolkit Configuration

To simplify the migration of the JumpStart environment between sites, specific configuration information is kept in configuration files. The Toolkit has two configuration files, which are both stored in the `Drivers` directory. The first,

`driver.init`, should rarely require modification. All site-specific and installation-specific modifications to the environment variables of the Toolkit should be made in the `user.init` script. This script contains the following variables:

- JASS_FILES_DIR
- JASS_FINISH_DIR
- JASS_PACKAGE_DIR
- JASS_PACKAGE_MOUNT
- JASS_PATCH_DIR
- JASS_PATCH_MOUNT
- JASS_RHOSTS_FILE
- JASS_ROOT_DIR
- JASS_ROOT_PASSWD
- JASS_SAVE_BACKUP
- JASS_SGID_FILE
- JASS_STANDALONE
- JASS_SUFFIX
- JASS_SUID_FILE
- JASS_SVCS_DISABLE
- JASS_TMPFS_SIZE
- JASS_UMASK
- JASS_UNAME
- JASS_UNOWNED_FILE
- JASS_USER_DIR
- JASS_WRITEABLE_FILE

Only these environment variables need to be verified when moving the JumpStart environment from one site to another. The function of each variable is as follows:

## JASS_CONFIG_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_CONFIG_DIR`. In standalone mode, it will be set by the `jass-execute` script. Normally this variable should not require modification by the user.

## JASS_FILES_DIR

This variable points to the location of the `Files` directory on the JumpStart server. This directory contains files which can be copied to the JumpStart client.

Any files to be copied are specified in the `JASS_FILES_LIST` variable—these will be copied to the client during installation. The `JASS_FILES_LIST` 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 Part 3 of this series.

## JASS_FINISH_DIR

This variable should not normally require modification. 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.

## JASS_PACKAGE_DIR and JASS_PACKAGE_MOUNT

These variables specify where software packages are stored.

The `JASS_PACKAGE_DIR` variable specifies where to NFS mount the `JASS_PACKAGE_MOUNT` directory. Normally, the `JASS_PACKAGE_DIR` variable will not require modification because this is a transient mount-point used only during the JumpStart installation.

The `JASS_PACKAGE_MOUNT` variable identifies the location of software packages available for installation on the JumpStart server. The location must be specified by both hostname or IP address and complete path because this directory is NFS mounted to the JumpStart client during installation. Since a hostname or IP address is specified in the value of the environment variable, it will *always* require modification. This is a JumpStart specific variable, and is not used during standalone installations.

## JASS_PATCH_DIR and JASS_PATCH_MOUNT

These variables specify the location of the `Patches` directory on the JumpStart server.

The `JASS_PATCH_DIR` variable specifies the directory where the `Patch` directory will be mounted during a JumpStart installation and does not usually require modification.

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. This is a JumpStart specific variable, and is not used during standalone installations.

## JASS_RHOSTS_FILE

This variable specifies where the `print-rhosts.fin` 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_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 must be changed to one. Toolkit version 0.2 automates this in the `jass-execute` script so manual modification is no longer required.

### JASS_ROOT_PASSWD

This variable specifies the root password used by the `set-root-password.fin` script. This will only be executed when using the Toolkit in JumpStart mode because the `set-root-password.fin` script does not run when the Toolkit is run in standalone mode and the system is not being booted off of a mini-root.

### 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.

### JASS_SGID_FILE

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

### 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. This variable should not require direct modification by the user as the `jass-execute` script will set it for standalone mode execution. The default value of 0 is correct for JumpStart installations.

### 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.

## JASS_SUID_FILE

This variable specifies where the `print-suid-files.fin` 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_SVCS_DISABLE

This variable can be used to simplify the removal of different services for different servers. When specified, the list of services defined in this variable will be disabled by the `update-inetd-conf.fin` script. When not specified, the Toolkit will default back to the hard-coded list of services specified in the script itself.

## JASS_TMPFS_SIZE

This variable can be used to specify the maximum disk space which may be used in the `tmpfs` filesystem. By default, this is hard-coded in the `set-tmpfs-limit.fin` script to be 100MB. When this variable is defined, the value it specifies will be used instead of the default.

## JASS_UMASK

This variable provides an override capability for the `set-system-umask.fin` and `set-user-umask.fin` scripts. The default value is `022` but can be set to any value desired through the use of the `JASS_UMASK` environment variable without modifying the script.

## JASS_UNAME

This variable is used as a global environment variable specifying the OS version of the JumpStart 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.

## JASS_UNOWNED_FILE

This variable specifies where the `print-unownded-files.fin` 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_USER_DIR

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

### JASS_WRITEABLE_FILE

This variable specifies where the output of the `print-world-writeable-files.fin` 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 have the output directed to standard output.

## Limitations

JumpStart is an extremely powerful tool; however, it does have limitations and restrictions. For instance, while booting, a JumpStart client will load a Solaris OE mini-root and run all subsequent commands from this memory based operating system. The operating system being installed is mounted on the mini-root through the mountpoint `/a`. However, many of the required commands can only be run on the disk-based OS and not from the memory resident mini-root. These commands and scripts must be called through the `chroot` command. By using the `chroot` command, the commands and scripts can be run on the newly installed OS image of the client system.

## Version Control

Maintaining version control for all files and scripts in the JumpStart environment is critical for two reasons. First, one of the goals of this environment is to be able to re-create a system installation. This will be impossible without having a snapshot of all file versions used during the installation. Secondly, because these scripts are performing security functions—which is a critical process for many organizations—extreme caution should be exercised to ensure only appropriate and tested changes are implemented.

The Source Code Control System (SCCS) used for version control is contained in the Solaris OE `SUNWsprot` package. Other version control software available from freeware and commercial vendors can also manage version information. Whichever version control product is used—it is important that a process *be in place* to manage updates and capture version information for future system re-creation.

# Toolkit (Part 3)

The following article will present detailed information on Toolkit installation and configuration. Additionally, the scripts contained in the Toolkit will be individually listed and discussed. Recommendations on changes necessary to port the Toolkit will also be made.

Part 3 of this series is available at:

`http://www.sun.com/blueprints/1100/jssec3-updt1.pdf`

# Conclusion

This article provided an overview of the JumpStart Architecture and Security Scripts Toolkit. As part of the overview, the design philosophy of the Toolkit was also reviewed. Additionally, the architecture and framework of the Toolkit was discussed. As part of the architecture discussion, the directory structures and their functions were described.

# Bibliography

*Solaris Advanced Installation Guide*, Sun Microsystems,
`http://docs.sun.com`

Dik, Casper, *fix-modes tool*,
`ftp://ftp.wins.uva.nl/pub/solaris/fix-modes.tar.gz`

Noordergraaf, Alex , *JumpStart Architecture and Security Scripts for the Solaris Operating Environment - Part 1*, Sun BluePrints OnLine, July 2000,
`http://www.sun.com/blueprints/0700/jssec.pdf`

Noordergraaf, Alex, *JumpStart Architecture and Security Scripts for the Solaris Operating Environment - Part 2*, Sun BluePrints OnLine, August 2000,
`http://www.sun.com/blueprints/0800/jssec2.pdf`

Noordergraaf, Alex, *Solaris Operating Environment Minimization for Security: Updated for Solaris 8*, Sun BluePrints OnLine, November 2000,
`http://www.sun.com/blueprints/1100/minimization-updt1.pdf`

Noordergraaf, Alex and Watson, Keith, *Solaris Operating Environment Minimization for Security*, 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 1999,
`http://www.sun.com/blueprints/1299/network.pdf`

Noordergraaf, Alex and Watson, Keith, *Solaris Operating Environment Security*, Sun BluePrints OnLine, January 2000,
`http://www.sun.com/blueprints/0100/security.pdf`

Powell, Brad, et. al., *Titan Toolkit*,
`http://www.fish.com/titan`

# Acknowledgements

I would like to thank Keith Watson for his input and assistance in the development and testing, and for the many hours spent reviewing and editing.

Keith Watson reviewed the end result of the Toolkit and made recommendations on needed changes, and also developed some elegant `Finish` scripts.

## *Author's Bio: Alex Noordergraaf*

*Alex Noordergraaf has over 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 SunPS. His customers have included major telecommunication firms, financial institutions, ISPs, and ASPs.*

## *Author's Bio: Glenn Brunette*

*Glenn Brunette has over 8 years experience in the areas of computer and network security. Glenn currently works with in the Sun Professional Services organization where he is the Lead Security Architect for the North Eastern USA region. 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, ISP, New Media, and government organizations.*

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