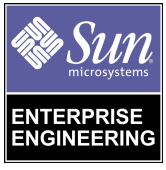
Building and Deploying OpenSSH for the Solaris™ Operating Environment

---

By Jason Reid and Keith Watson

Sun BluePrints™ OnLine - July 2001



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.: 816-1454-10
Revision 01, July 2001

Please
Recycle

Building and Deploying OpenSSH for the Solaris™ Operating Environment

The network environment has never been truly safe, and it is getting less safe with each passing day. As more network users require greater access to remote systems, the risk of compromised accounts and systems increases. Most users prefer to use the network access tools they are most familiar with such as `telnet`, `ftp`, `rlogin`, `rsh`, and `rcp`. Unfortunately, these tools do not provide mechanisms for strong authentication or privacy through encryption. This article describes how to build and deploy OpenSSH for the Solaris™ Operating Environment (Solaris OE) to enable secure remote network connections with strong authentication and encryption.

Attackers have a wide arsenal of tools at their disposal to capture user names and passwords as they cross the network. Once the attacker captures a user name and password from one of these unsafe network tools, that user account (or possibly, administrator account) is compromised. Since the network session is transferred in clear-text, attackers can also eavesdrop on a session and take it over (known as hijacking), or apply one of several other serious network attacks.

Fortunately, there are tools available for the Solaris OE that provide protection against attacks and provide similar replacements for common network access tools. OpenSSH is one of several tools that implement the SSH1 and SSH2 protocols to provide the necessary network protection. OpenSSH encrypts all network traffic, provides stronger authentication, and monitors the integrity of the network session. It also provides a tunneling mechanism for X-windows communications and other unsafe network services. OpenSSH is an open source tool, built from several other open source components, and is available for free with no license restrictions or patented algorithms. Most importantly, OpenSSH provides equivalent replacements for the commands that users are familiar with. This allows an organization to quickly switch to OpenSSH with little user training required. OpenSSH compiles and runs on the Solaris OE and can be deployed immediately.

OpenSSH is managed by the OpenBSD team. OpenBSD is an open source operating system based on BSD 4.4-Lite and is available for free. A major goal of the OpenBSD project is to create a secure operating system by auditing source code, fixing security problems quickly, and integrating security tools and cryptographic software. To achieve these goals, the OpenBSD team created the OpenSSH project. OpenSSH is tightly integrated with OpenBSD. An OpenSSH porting team makes the code available for other platforms, such as the Solaris OE. The “portable” version of the OpenSSH code is typically available at the same time or a few days after the main release.

Several components must be built prior to building OpenSSH itself. This article describes each necessary component and provides recommendations for configuration and compilation options. OpenSSH is a flexible tool with several options that affect its integration into a site’s security policy. These options are explored in this article, and issues of packaging and deployment are also addressed.

The compilation of OpenSSH and its components described in this article have only been tested on the following releases:

- Solaris 2.6 5/98 OE for SPARC™
- Solaris 7 11/99 OE for SPARC
- Solaris 8 4/01 OE for SPARC

with the following compilers:

- Forte™ C 6 Update 1 (formerly Sun Workshop™ Compilers C)
- GNU Compiler Collection (gcc) 2.95.2

OpenSSH and its components are constantly evolving. New versions of OpenSSH and the other software components discussed in this article become available frequently and with little fanfare. The versions used for this article may not be the most current available. Always check the Web sites listed in the *Bibliography* section to ascertain the software versions available at the time you build OpenSSH.

The use of cryptographic tools is strictly regulated or even prohibited in some areas of the world. It is recommended to seek advice from your legal counsel before deploying OpenSSH if you are unsure of the local legal requirements.

OpenSSH and Its Components

The Solaris OE does not include many of the integrated components of OpenBSD. These must be compiled separately for the Solaris OE and linked with the final OpenSSH executables. All of the components needed are open source tools with no restrictive licensing requirements. The following are the components needed and URLs to where the code can be found:

- OpenSSH 2.9p2 (required) `http://www.openssh.com/portable.html`
  This is the core code for OpenSSH. Be sure to download the "portable" version of the code.
- Zlib 1.1.3 (required) `http://www.freesoftware.com/pub/infozip/zlib/`
  This is an open source library that contains the data compression algorithms that OpenSSH uses.
- OpenSSL 0.9.6b (required) `http://www.openssl.org/source/`
  This is an open source library that contains the cryptographic algorithms that OpenSSH uses.

Note – The OpenSSL library contains patented cryptographic algorithms; however, OpenSSH does not use them.

- PRNGD 0.9.19 (optional)
  `http://www.aet.tu-cottbus.de/personen/jaenicke/postfix_tls/prngd.html`
  This is an open source entropy gathering and collection tool that OpenSSH uses to generate random numbers used in cryptographic operations.

The pseudo random number generator daemon (PRNGD) is listed as optional because OpenSSH can perform entropy collection on its own. However, OpenSSH executables pause for a period of time to gather entropy prior to cryptographic operations. PRNGD provides a constantly filled entropy pool that OpenSSH can draw from without pausing for any significant time. Unfortunately, PRNGD is an additional executable to install and runs as a separate process which must be started at boot. It is addressed in the provided `init` script discussed later in this article.

Building the Components

The commands and options are listed for each component discussed in the following sections. In some cases, the default actions are not appropriate for the Solaris OE. The following sections document the default options with the correct options and include an explanation for the deviation. Also, the commands listed have been

tested with Korn Shell 88 (/usr/bin/ksh) only. Compiling code is typically an output-intensive operation, so the build and compiler output is not included in this article.

To build these components and OpenSSH, the build system should have the Developer cluster installed for the appropriate release of the Solaris OE. The Developer cluster contains the necessary packages, libraries, header files, and executables to compile source code. If you are deploying OpenSSH to older releases of the Solaris OE, it is necessary to build it on a system with the oldest release. For example, an OpenSSH executable built on a Solaris 8 OE system may not operate on a Solaris 2.6 OE system. However, newer releases of the Solaris OE are able to execute OpenSSH executables built for an older release. Always build on the oldest Solaris OE release that is supported throughout the environment.

For the choice of compilers, both Forte and GNU are sufficient. However, the Forte compiler generally produces more optimized executables. Fast and efficient execution is necessary for cryptographic operations to generate acceptable performance. Make sure the build system has the specified patches to assure correct performance—this is particularly critical for older versions of the Forte compiler.

Each of these components install their libraries in the /usr/local directory. These libraries are needed on the development system during the build process. It is possible to specify different directories for installation. However, these new directories will need to be specified to the OpenSSH configure script because it searches for the libraries in the /usr/local directory by default. Also, the OpenSSH executables link to the component libraries during the build process. There is no requirement for these libraries to be installed on each system that uses OpenSSH.

To build the components and OpenSSH, be sure to add the appropriate directories to your shell search path. The /usr/ccs/bin directory path is required in addition to the path to the compiler. For example, the following commands set the correct path:

sh/ksh:

```
$ PATH=/opt/SUNWspro/bin:/usr/ccs/bin:$PATH
```

csh:

```
% set path = ( /opt/SUNWspro/bin /usr/ccs/bin $path )
```

It is also recommended to add `/usr/ccs/bin` to the search path after becoming root to install the libraries.

Building Zlib

To configure Zlib to use the gcc compiler, enter the following commands:

```
$ cd zlib-1.1.3
$ ./configure
```

To configure Zlib to use the Forte compiler, enter the following commands:

```
$ cd zlib-1.1.3
$ env CC=cc CFLAGS="-xO4 -KPIC" ./configure
```

To build and install Zlib, enter the following commands:

```
$ make
$ su
Password: password
# PATH=$PATH:/usr/ccs/bin
# export PATH
# make install
# ls -l /usr/local/lib/libza.a
-rwxr-xr-x   1 root     other      65984 May 18 19:56 libz.a*
```

Building OpenSSL

The `Configure` script for OpenSSL attempts to build a library that is optimized for a specific hardware architecture. If OpenSSH is to run on many different SPARC platforms, OpenSSL should be compiled more generally to accommodate the different SPARC architectures. TABLE 1 lists the supported architectures and their associated designations used by the OpenSSL `Configure` script based on compiler type.

TABLE 1 Supported Architectures and Associated Designations

Supported Architectures	Forte Workshop Compiler (cc)	GNU Compiler Collection (gcc)
sun4c, sun4d, sun4m, sun4u	solaris-sparcv7-cc	solaris-sparcv7-gcc
sun4d, sun4m	solaris-sparcv8-cc	solaris-sparcv8-gcc
sun4u	solaris-sparcv9-cc	solaris-sparcv9-gcc solaris-sparcv9-gcc27 (Version 2.7 of gcc)
i86pc (Intel)	Not supported	solaris-x86-gcc

Note – There are also the solaris-sparc-sc3 and solaris64-sparcv9-cc designations. The solaris-sparc-sc3 designation is for C SPARCompiler 3.0 which is neither supported nor recommended for the 2.6, 7, and 8 versions of the Solaris OE. The solaris64-sparcv9-cc designation is for building a 64-bit version. Do not use this designation because it will not link with the 32-bit components of OpenSSH and Zlib.

Note – Regardless of the version of Forte C, apply the necessary patches for the compiler. OpenSSL requires patched versions to operate correctly. Refer to `http://access1.sun.com/` for the necessary patches.

Determine all of the architectures that your deployment of OpenSSH must support. Select the designation for the particular compiler you intend to use. That designation must be specified as the argument to the OpenSSL `Configure` script.

Note – Perl 5 is required by the OpenSSL `Configure` script and the OpenSSH `configure` script. Perl 5 is included with the Solaris 8 OE but not previous releases.

Enter the following commands to configure OpenSSL using the `Configure` script and your selected compiler designation:

```
$ cd openssl-0.9.6b
$ ./Configure designation
```

To build and install OpenSSL library, enter the following commands:

```
$ make
$ su
Password: password
# PATH=$PATH:/usr/ccs/bin
# export PATH
# make install
# ls -l /usr/local/ssl/lib
total 4976
-rw-r--r--   1 root     other    2161844 May 18 20:50 libcrypto.a
-rw-r--r--   1 root     other     367704 May 18 20:50 libssl.a
```

Building PRNGD

There is no configure script for PRNGD.

To build PRNGD using the gcc compiler, enter the following commands:

```
$ cd prngd-0.9.19
$ make CC=gcc CFLAGS="-O3 -DSOLARIS" SYSLIBS="-lsocket -lnsl"
```

To build PRNGD using the Forte C compiler, enter the following commands:

```
$ cd prngd-0.9.19
$ make CC=cc CFLAGS="-xO4 -DSOLARIS -KPIC" SYSLIBS="-lsocket -lnsl"
```

Note – Check the `prngd-0.9.19` `Makefile` for the appropriate `CFLAGS` to use on releases previous to Solaris 7 OE.

To install PRNGD, enter the following commands:

```
$ su
Password: password
# cp prngd /usr/local/sbin/prngd
# chown root:bin /usr/local/sbin/prngd
# chmod 755 /usr/local/sbin/prngd
# cp contrib/Solaris-7/prngd.conf.solaris-7 /etc/prngd.conf
# cat /var/log/syslog > /etc/prngd-seed
```

Building OpenSSH

All the required and optional components must be built and installed before configuring OpenSSH.

It is recommended to build OpenSSH with the following settings:

- `--with-pam` (Enables use of the Pluggable Authentication Module (PAM) architecture for Solaris OE.)
- `--disable-suid-ssh` (Installs the `ssh` command without the set-UID bit. This will prevent a root compromise if a vulnerability is found in the `ssh` command. A side effect is that `.rhosts` authentication is disabled.)
- `--sysconfdir=`*`DIRECTORY`* (Places the OpenSSH configuration files and cryptographic keys in the specified directory. By default on the Solaris OE, these files are stored in the `/usr/local/etc` directory.)
- `--with-prngd-socket=`*FILE* (Enables optional PRNGD support. Provide the filename for socket file.)
- `--without-rsh` (Prevents fallback to insecure `rsh` in case of a failure to connect through SSH1 or SSH2 protocols.)
- `--prefix=`*`DIRECTORY`* (Specifies the top-level OpenSSH installation directory.)

Note – The Solaris OE software package creation script described later in the article installs OpenSSH in the `/opt/OBSDssh` directory. This is configurable; however, the top-level directory specified when configuring OpenSSH must match the installation directory in the `makeOpenSSHPackage.ksh` script. The examples listed below use the `/opt/OBSDssh` directory.

To configure OpenSSH to use the gcc compiler and PRNGD, enter the following command:

```
$ ./configure --prefix=/opt/OBSDssh --with-pam --without-rsh \
--disable-suid-ssh --sysconfdir=/etc \
--with-prngd-socket=/var/spool/prngd/pool
```

To configure OpenSSH to use the Forte C compiler and PRNGD, enter the following command:

```
$ env CC=cc ./configure --prefix=/opt/OBSDssh --with-pam \
--without-rsh --disable-suid-ssh --sysconfdir=/etc \
--with-prngd-socket=/var/spool/prngd/pool
```

To configure OpenSSH to use the gcc compiler and *NOT* use PRNGD, enter the following command:

```
$ ./configure --prefix=/opt/OBSDssh --with-pam --without-rsh \
--disable-suid-ssh --sysconfdir=/etc
```

To configure OpenSSH to use the Forte C compiler and *NOT* use PRNGD, enter the following command:

```
$ env CC=cc ./configure --prefix=/opt/OBSDssh --with-pam \
--without-rsh --disable-suid-ssh --sysconfdir=/etc
```

To build OpenSSH, enter the following command:

```
$ make
```

To install OpenSSH, enter the following command:

```
$ su
Password: password
# PATH=$PATH:/usr/ccs/bin
# export PATH
# make install
# ls -l /opt/OBSDssh/bin/ssh
-rwxr-xr-x   1 root     other    1451384 May 18 22:14 ssh
```

Deploying OpenSSH and its Components

In order to simplify the installation and operation of OpenSSH for the Solaris OE, several scripts have been created in conjunction with this article. These are available at the Sun BluePrints™ OnLine Scripts and Tools Web page.

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

For simpler deployment of OpenSSH, it is recommended that a Solaris OE software package be created. This allows addition and removal of packages, and facilitates installation with JumpStart™ technology. The `makeOpenSSHPackage.ksh` script builds the `OBSDssh` package in Solaris OS package stream format. Execute `makeOpenSSHPackage.ksh` from within the `openssh-2.9p2` directory.

An `init` script is provided to manage to the automatic start of the OpenSSH server process at system boot. That script is named `openssh.server`.

A JumpStart installation script for the OpenSSH Solaris OE software package (created by the `makeOpenSSHPackage.ksh` script) is also included. This scripts is named `install-openssh.fin` and requires the Solaris™ Security Toolkit. This Toolkit is also available from the Sun BluePrints OnLine Scripts and Tools Web page.

Client and Server Configuration Notes

Prior to installing or deploying OpenSSH, examine the configuration of the OpenSSH client and server. Altering the configuration before deployment saves time later if changes must be made. The OpenSSH client configuration file is located in the top-level directory of the OpenSSH source tree and named `ssh_config` (edit the file named `ssh_config.out`). The OpenSSH server configuration file is located in the same place and named `sshd_config` (edit the file named `sshd_config.out`). The `makeOpenSSHPackage.ksh` script copies the `ssh_config.out` and `sshd_config.out` files into the Solaris OE software package that is created.

Solaris OE Package Creation

The `makeOpenSSHPackage.ksh` script is provided to create a Solaris OE software package. It takes the OpenSSH executables (and optionally the PRNGD executable and associated files), configuration files, and manual pages, and creates the package. This script may need to be manually edited for changes specific to the environment in which OpenSSH is to be installed. Be sure that the user configuration in the `makeOpenSSHPackage.ksh` script matches the OpenSSH compilation options. Also, the `makeOpenSSHPackage.ksh` script alters the configuration of the `openssh.server` script based on the location of the OpenSSH executables and configuration files as defined in the `makeOpenSSHPackage.ksh` script.

To build the OBSDssh package, enter the following commands after verifying the configuration of the makeOpenSSHPackage.ksh script:

```
$ cp makeOpenSSHPackage.ksh openssh-2.9p2
$ cd openssh-2.9p2
$ chmod +x makeOpenSSHPackage.ksh
$ ./makeOpenSSHPackage.ksh
$ ls -l OBSDssh.pkg
-rw-r--r--   1 kaw      kaw      7399424 May 20 17:22 OBSDssh.pkg
```

To install the OBSDssh package, enter the following commands:

```
$ su
Password: password
# /usr/sbin/pkgadd -d OBSDssh.pkg OBSDssh
```

To remove the OBSDssh package, enter the following commands:

```
$ su
Password: password
# /usr/sbin/pkgrm OBSDssh
```

init Script

An init script is required for the automatic start of both the sshd and PRNGD daemons. The init script allows for a standard method of starting and stopping a service. There are also some housekeeping issues involved. The sshd daemon requires initial key generations and PRNGD requires initial seed generations. The openssh.server script takes care of these issues. The installation of this script is included with the installation of the OBSDssh package.

To start OpenSSH, enter the following commands:

```
$ su
Password: password
# /etc/init.d/openssh.server start
```

To stop OpenSSH, enter the following commands:

```
$ su
Password: password
# /etc/init.d/openssh.server stop
```

Solaris Security Toolkit Software Installation

In order to facilitate the deployment of OpenSSH for the Solaris OE, a finish script for the Solaris Security Toolkit has been provided. The Toolkit automates and simplifies building secured Solaris OE systems based on the recommendations contained in several security-related Sun BluePrints articles. The Toolkit focuses on Solaris OE security modifications to harden and minimize a system.

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. Additionally, the Toolkit can also be run outside the JumpStart framework in a standalone mode. This standalone mode allows the Toolkit to be used on systems that require security modifications or updates but cannot be taken out of service to reinstall the OS from scratch.

Sun BluePrints articles describing this security Toolkit are available at:

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

The Solaris Security Toolkit itself can be downloaded from:

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

The `install-openssh.fin` script provides a simple way to deploy OpenSSH to Solaris OE systems during network installation. This script and the OpenSSH Solaris OE software package must be copied onto the JumpStart server and added to the appropriate configuration files. See the Solaris Security Toolkit documentation for more details.

Conclusion

The public network is not safe. Insecure protocols such as `telnet` allow passwords to be snooped and sessions hijacked. Cryptographic solutions such as OpenSSH exist to mitigate this problem. The deployment of OpenSSH requires planning and testing but should be applicable to most environments. Being aware of the known attacks will hopefully provide the needed leverage to deploy OpenSSH.

Using the `openssh.server`, `makeOpenSSHPackage.ksh`, and `install-openssh.fin` scripts can assist you in easily and rapidly deploying OpenSSH in your enterprise. OpenSSH provides a familiar interface to users and reduces their retraining needs.

Consider disabling unsafe network services such as `telnetd`, `ftpd`, `rlogind`, and `rshd` after the deployment of OpenSSH.

Bibliography

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

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

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

Forte Workshop C, `http://www.sun.com/forte/`

OpenBSD, `http://www.openbsd.org/`

OpenSSH, `http://www.openssh.com/`

OpenSSL, `http://www.openssl.org/`

PRNGD, `http://www.aet.tu-cottbus.de/personen/jaenicke/postfix_tls/prngd.html`

Solaris Operating Environment and Forte Workshop patches, `http://sunsolve.sun.com/` and `http://access1.sun.com/`

Zlib, `http://www.freesoftware.com/pub/infozip/zlib/`

Author's Bio: Keith Watson

Keith Watson has spent nearly four years at Sun working in the area of computer and network security. He is currently the product manager for core Solaris security. Previously, Keith was a member of the Global Enterprise Security Service (GESS) team in Sun Professional Services. He is also a co-developer of an enterprise network security auditing tool named the Sun Enterprise Network Security Service (SENSS). Prior to joining Sun, Keith was part of the Computer Operations, Audit, and Security Technologies (COAST) laboratory (now part of the CERIAS research center) at Purdue University.

Author's Bio: Jason Reid

Jason Reid is a test engineer in the Solaris System Test Group. He has also been an SQA engineer in the Developer Tools Group. Prior to joining Sun, Jason worked at the Purdue University Computing Center as an UNIX® systems administrator, while obtaining his BS in computer science.