Configuring the Secure Shell Software

Jason Reid, Solaris™ System Test

Sun BluePrints™ OnLine—April 2003

http://www.sun.com/blueprints

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

Part No. 817-2485-10
Revision 06, 4/7/03
Edition: April 2003

Please
Recycle

Configuring the Secure Shell Software

Networks have never been secure. As the demand on open networks for remote access has grown, the risks of compromised systems and accounts has kept pace. Tools for securing networks, such as the Secure Shell software, were developed to counter the threats of password theft, session hijacking, and other network attacks. However, these tools need to be properly configured to the local environment.

This article provides recommendations on configuring two specific Secure Shell implementations for the Solaris™ Operating Environment (Solaris OE): OpenSSH and the Solaris™ Secure Shell software. The Solaris Secure Shell software is a component of the Solaris 9 OE release. For previous Solaris OE releases, OpenSSH is available. For information on building OpenSSH, consult the Sun BluePrints™ OnLine article “Building OpenSSH—Tools and Tradeoffs” (January 2003).

This article updates the configuration recommendations provided in the Sun BluePrints OnLine article “Configuring OpenSSH for the Solaris Operating Environment” (January 2002).

Security Policy

The primary purpose of security policy is to inform those responsible for protecting assets, such as hardware, software, and data, of their obligations. Management establishes the policy based on the risks it is willing to tolerate. The policy itself does not set goals but serves as a bridge between management’s goals and the technical implementation. Configuration is the technical implementation.

The policy should guide the configuration. If you do not have a policy, I strongly recommend that you establish one. To help you develop a policy, refer to the Sun BluePrint OnLine article “Developing a Security Policy” (December 2001). There are a variety of resources available to assist in setting policy.

In configuring Secure Shell software, keep in mind two principles:

- Defend-in-depth

  Let no single point of configuration or defense be the only gatekeeper for security.

- Plan on failure

  Secure Shell software can, and should, be configured at multiple points (build-time, server configuration, and client configuration). No single misconfiguration should completely break the system security.

This article presents example client and server configurations in “Appendix: Configuration Files” on page 12. These example configurations are for a variety of environments ranging from a workstation to a DMZ bastion host, but not all of the options presented in these examples are described in this document. For more information, consult the vendor documentation.

Configuration

In order of precedence, Secure Shell configuration occurs at the following places:

- Software build-time
- Server command line options
- Server configuration file (`sshd_config`)
- Client command line options
- User client configuration file (`~/.ssh/config`)
- Global client configuration file (`ssh_config`).

Build time configuration is the strongest configuration type. It cannot be changed without rebuilding the software. This is inconvenient if a change is needed. The build-time configuration of the Solaris Secure Shell software is fixed and cannot be changed.

The server configuration involves how the `sshd`(1M) daemon will present itself on the network, what protocols and authentication methods are acceptable, and how the user environment is constructed. The client configuration involves determining which server to transact with which protocol, verifying the server identity,

determining the user identity presentation, and choosing the ease-of-use features. Policy details are implemented on the server side. The client cannot override, nor provide, a feature that the server does not offer.

The available features can be enabled or disabled by either command-line options or the applicable configuration file. Command-line options apply to that particular instantiation of either the server or client. Configuration-file options are persistent until the file is altered and a new instantiation started. The most reliable configuration method uses the configuration file. This gives a repeatable, reproducible invocation. Changes can also be tracked by using source control. For information on command-line options, consult the vendor documentation.

Mechanics of Configuration Files

OpenSSH places `sshd_config` and `ssh_config` in the location specified by the `sysconfdir` keyword when OpenSSH is built. The usual values are `/etc`, `/usr/local/etc`, `/etc/ssh`, or `/etc/openssh`. The Solaris Secure Shell software stores the two files in `/etc/ssh`. These files should be owned by user `root` and group `sys`. The file permission mode should be either 644 or 444. The server checks for the optional user configuration file at `~/.ssh/config`. This file should not be world or group writable.

Configuration files contain two types of entries: comments and keyword-value pairs. Comments are blank lines and lines beginning with the hash mark (#). Keyword-value pairs consist of an identifier (keyword), a space, then the value associated with the identifier. Keywords are case-insensitive while values are case-sensitive.

Traditionally, keywords have the first letter of each word capitalized for readability. Some values are lists that are either comma or space delimited depending on the keyword. Consider keeping configuration files under source control to track revisions. The source control tags can be hidden by the comment character (the hash mark).

```
# Example config file - two comments and one
# keyword-value pair
Port 22
```

Recommendations

During configuration, trade-offs are made between security, ease-of-use, and legacy compatibility. There are a wide variety of options covering network and protocol support, authentication, and user environment, that obscure the individual option's impact to the whole. This section includes some recommendations, along with the consequences of their usage.

Note – Only the Solaris Secure Shell software (Solaris 9 OE) and OpenSSH (3.5p1) versions that were current at the time this article was written were used. Not all of the options are covered. Consult the vendor documentation for more information on the other options and on the options presented here.

Server Recommendations

Server configuration concerns how the daemon presents itself on the network, what protocols are offered, and what authentication methods are allowed. Specific recommendations are given for each topic. Recommendations specific to a particular Secure Shell implementation have been noted.

Protocol Support

Two major versions of the Secure Shell protocol exist. Protocol 1 has been deprecated due to vulnerabilities, such as packet insertion and password length determination. Whenever possible, use Protocol 2. Unfortunately, many legacy clients support only Protocol 1. If this protocol must be enabled, consult the Legacy Support recommendations later in this article. Consider migrating to clients that support Protocol 2, as soon as reasonably possible.

Network Access

By default, the `sshd`(1M) daemon listens on all network interfaces on its bound ports. For workstations or other systems where accessibility is desired on all interfaces, this is not a problem. For architectures such as the Service Delivery

Network, in which management traffic is limited to a particular interface, it is a problem. Limit network access with the `ListenAddress` keyword, where access is limited by a particular IP address, not a network interface.

```
# Listen only to the management network.
ListenAddress 192.168.0.10
```

To further narrow down what the daemon will listen to, use either a host-based firewall, such as the SunScreen™ software, or TCP Wrappers.

For a reference on traffic limited architectures, consult the Sun BluePrints OnLine article “Building Secure N-Tier Environments” (October 2000).

Keep Alives

Occasionally, connections are temporarily suspended when a route is downed, a machine crashes, a connection is hijacked, or a man-in-the-middle attack is attempted. TCP keep alives should be sent to detect any of these cases. The server will disconnect the connection if TCP keep alives fail and return allocated resources. Regular disconnects can aggravate users on faulty networks.

```
KeepAlive yes
```

Data Compression

Optionally, you can use compression on the encrypted data streams. This results in bandwidth savings for compressible data, such as interactive logins or log files, at the expense of more CPU resources. For uncompressible data, such as encrypted or compressed files, the extra CPU time is wasted and decreases performance. For singular Secure Shell sessions, this is not a problem. For a file server, the extra load could impact performance. In this case, turn compression off to prevent misconfigured clients from driving up the system load.

```
# Transfering ASCII data such as interactive logins or log files
Compression yes
```

```
# Transfering random data such as compressed or encrypted files
# Prevents performance issues and reduces CPU load
Compression no
```

Privilege Separation

Privilege separation is a feature only in OpenSSH. The `sshd`(1M) daemon is split into two parts: a privileged process to deal with authentication and process creation and an unprivileged process to deal with incoming network connections. After successful authentication, the privileged process spawns a new process with the privileges of the authenticated user. The goal is to prevent compromise from an error in the network facing process. Unfortunately, privilege separation is not entirely compatible with pluggable authentication modules (PAM), SunShield™ Basic Security Module (BSM) auditing. Also, some OpenSSH features are disabled. If this functionality is desired, consult the vendor documentation.

```
# OpenSSH only
UsePrivilegeSeparation no
```

Login Grace Time

The default login grace time is the time a connection can exist before successfully authenticating. The default of 600 seconds for the Solaris Secure Shell software and 120 seconds for later OpenSSH versions is too long. Reduce the time to sixty seconds.

```
LoginGraceTime 60
```

Password and Public Key Authentication

Passwords are not always appropriate. The stronger identity method might be required. An identity is a public-key cryptographic, two-factor authentication system. When passwords are deemed sufficient, do not allow empty passwords. They are trivial to guess.

```
PasswordAuthentication yes
PermitEmptyPasswords no
PubKeyAuthentication yes
DSAAuthentication yes
```

Superuser (root) Logins

Neither the Solaris Secure Shell software nor OpenSSH honor the values set in the /etc/default/login file. To prevent network superuser (root) logins, this must be explicitly denied. By default, the Solaris Secure Shell software denies superuser logins; while, OpenSSH allows them. This forces system administrators to log in as unprivileged users, then change users (su) to the superuser. Only if the system has no user accounts and the appropriate host protection is in place should you enable superuser logins.

```
PermitRootLogin no
```

Banners, Mail, and Message-of-the-Day

Some sites require that a banner be displayed after a user connects to a system, but before logging in. You can accomplish this with the Banner keyword. Set Banner to /etc/issue so that only one banner file exists for the entire system.

```
Banner /etc/issue
```

In the Solaris OE, the interactive login shell is expected to display the message-of-the-day (MOTD) and to check for new mail. Using some versions of OpenSSH, this causes the MOTD display and mail check to be done twice. Set these keywords to no to eliminate the duplication.

```
CheckMail no
PrintMotd no
```

Connection and X11 Forwarding

Secure Shell can tunnel TCP and X connections through encrypted connections established between the client and server. Tunneling the traffic is referred to as forwarding. The forwarding occurs at the application level and is not completely transparent to the applications being forwarded. The applications need some configuration to use the tunnel.

Note – Data is protected only while it is in the tunnel between the client and server. After that, it is normal network traffic in the clear.

Tunneled traffic bypasses firewalls and intrusion detection systems. Allowing connection (TCP port) forwarding enables remote users safer access to email or the corporate Web server. X forwarding allows system administrators to run GUI applications remotely, such as the Sun™ Management Center software. This might not be functionality you want your users setting up. You can inconvenience users by turning off the functionality, but as long as they have shell access, they can run their own forwarders. Use role-based access control (RBAC) to explicitly limit what you want your users to do in this case.

If port forwarding is enabled, disable `GatewayPorts` and educate your users. `GatewayPorts` allows machines, other than the client, to access the forwarded ports in the tunnel. This effectively bypasses any firewall usage. Again, users could run their own private forwarders on their client machines to defeat the server restrictions. Consider placing an intrusion detection sensor on the private network side of a Secure Shell bastion host to detect problem traffic.

```
AllowTCPForwarding yes
GatewayPorts no
X11DisplayOffset 10
X11Forwarding yes
XAuthLocation /usr/X/bin/xauth
```

User Access Control Lists

User access control lists (ACL) can be specified in the server configuration file. You can either specifically allow or deny individual users or groups. No other part of the Solaris OE honors this ACL. The default is to allow access to anyone with a valid account. This can be a method to limit particular user's access in NIS environments, without resorting to custom pluggable authentication modules. Use only one of the following four ACL keywords (`AllowGroups`, `AllowUsers`, `DenyGroups`, or `DenyUsers`) in the server configuration file.

```
# Allow only the sysadmin staff
AllowGroups staff
```

```
# Or prevent unauthorized users.
DenyUsers Cheng Atkinson
```

User File Permissions

If users have left their home directories or `.ssh` files world writable by accident or trickery, an intruder could insert identities allowing password free access or alter the `known_hosts` file, allowing man-in-the-middle attacks. Enabling `StrictModes`, the `sshd`(1M) daemon will not allow a login. When on, users can be easily confused because they will not know why they cannot log in. No different error message is presented to them. For sites that will disable `StrictModes`, consider eliminating public-key-based authentication to prevent user account compromise. The consequence is the elimination of password-free logins for users or automated jobs.

```
StrictModes yes
```

`UseLogin` Keyword

For OpenSSH only, `UseLogin` specifies that the OpenSSH `sshd`(1M) daemon call `login`(1) instead of performing the initial login tasks itself for interactive sessions. This feature is used to work around OpenSSH's lack of support for SunShield BSM auditing. Enabling this feature should allow interactive users to be audited and prevent corruption of the user's cron audit file. This will not affect non-interactive (remote job execution) usage auditing, which will still not work. This feature disables X11 and port forwarding and is not compatible with privilege separation.

```
UseLogin no
```

Legacy Support

If legacy clients must be supported, strengthen the default configuration as much as possible. Default to Protocol 2 for the clients that support it. Disable all of the various `rhosts` style authentication methods. Increase the server key size and decrease the ephemeral key regeneration interval to minimize offline factoring attacks against the keys.

```
# Enable protocol 1 but default to protocol 2.
Protocol 2,1
# Legacy support options - protocol 1 only
HostKey /etc/ssh/ssh_host_key
IgnoreRhosts yes
IgnoreUserKnownHosts yes
KeyRegenerationInterval 1800
RhostsAuthentication no
RhostsRSAAuthentication no
RSAAuthentication yes
ServerKeyBits 1024
```

Client Recommendations

Client configuration concerns protocol support, identity management, and host options assignment. Specific recommendations are given for each topic.

Host Option Assignment

Configuration options can be assigned to a specific host or to all hosts by using the `Host` keyword. The value is matched to what the user types on the command line, not the actual hostname of the server. An asterisk (*) is used to set global default options. Options assigned to a specific host have precedence over the global default options.

```
# Only for a specific host
Host legacy
Protocol 1

# For all hosts
Host *
Protocol 2
```

Data Compression

Data compression can be used on the encrypted data stream to save bandwidth. Set to `off` by default, you should enable it for interactive sessions or for transferring easily compressible data. The compression cost is asymmetric in that compressing the data is more computationally expensive than decompression. Client-side CPU cycles are generally cheaper than server-side CPU cycles. Avoid attempting to compress already compressed or encrypted data to avoid needlessly raising the CPU load on the server.

```
# For interactive sessions, low bandwidth links,
# or easily compressable files
Compression yes
```

Keep Alives

Enable TCP keep alives to detect downed connections. See “Keep Alives” on page 5 for the server recommendations.

```
KeepAlive yes
```

Protocol Support

Always use Protocol 2 when possible. See “Protocol Support” on page 4 for the server recommendation.

```
Protocol 2
```

`rlogin` and `rsh`

The `rlogin` and `rsh` protocols should not be used. Prevent the client from attempting to execute `rsh` in case a Secure Shell connection is refused.

```
FallBackToRsh no
UseRsh no
```

Server Identity

Verify server identity both by its host key and IP address. For higher levels of identity assurance, set `StrictHostKeyChecking` to `yes` and distribute hosts keys out-of-band. This is impractical when users frequently encounter new hosts. Set `StrictHostKeyChecking` to `ask`, and train the users to verify the offered host key with the stored host key on the server.

```
CheckHostIP yes
# only access one host
StrictHostKeyChecking yes
# access a variety of hosts
StrictHostKeyChecking ask
```

Appendix: Configuration Files

This section contains examples of server and client configuration files.

Server Configuration Files

This section contains examples of server configuration files that you can use in your environment.

DMZ-Bastion Host Server

The following is an example of the DMZ-bastion host server Secure Shell configuration file:

```
# Protocol and server operation
Compression yes
KeepAlive yes
MaxStartups 10
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
Protocol 2
Port 22
# If using OpenSSH
UseLogin no
```

```
UsePrivilegeSeparation no

# Authentication
# Only allow public key based authentication. No passwords.
DSAAuthentication yes
LoginGraceTime 60
PAMAuthenticationViaKBDInt yes
PasswordAuthentication no
PermitEmptyPasswords no
PermitRootLogin no
PubKeyAuthentication yes

# User environment
AllowTCPForwarding no
Banner /etc/issue
CheckMail no
GatewayPorts no
PrintMotd no
StrictModes yes
X11Forwarding no
```

Legacy Support

The following is an example of the Secure Shell server configuration file with legacy support.

```
# Protocol and server operation
Compression yes
KeepAlive yes
MaxStartups 10
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
# Enable protocol 1 but default to protocol 2.
Protocol 2,1
Port 22
# If using OpenSSH
UseLogin no
UsePrivilegeSeparation no

# Authentication
DSAAuthentication yes
LoginGraceTime 60
PAMAuthenticationViaKBDInt yes
PasswordAuthentication yes
PermitEmptyPasswords no
PermitRootLogin no
```

```
PubKeyAuthentication yes

# User environment
AllowTCPForwarding yes
Banner /etc/issue
CheckMail no
GatewayPorts no
PrintMotd no
StrictModes yes
X11DisplayOffset 10
X11Forwarding yes
XAuthLocation /usr/X/bin/xauth

# Legacy support options - protocol 1
HostKey /etc/ssh/ssh_host_key
IgnoreRhosts yes
IgnoreUserKnownHosts yes
KeyRegenerationInterval 1800
RhostsAuthentication no
RhostsRSAAuthentication no
```

Workstation Server

The following is an example of the Secure Shell workstation server configuration file:

```
# Protocol and server operation
Compression yes
KeepAlive yes
MaxStartups 10
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
Protocol 2
Port 22
# If using OpenSSH
UseLogin no
UsePrivilegeSeparation no

# Authentication
DSAAuthentication yes
LoginGraceTime 60
PAMAuthenticationViaKBDInt yes
PasswordAuthentication yes
PermitEmptyPasswords no
PermitRootLogin no
PubKeyAuthentication yes
```

```
# User environment
AllowTCPForwarding yes
Banner /etc/issue
CheckMail no
GatewayPorts no
PrintMotd no
StrictModes yes
X11DisplayOffset 10
X11Forwarding yes
XAuthLocation /usr/X/bin/xauth
```

Client Configurations

This section contains examples of the client configuration files for remote workers and client workstations.

Remote Worker Configuration File

The following is an example of the Secure Shell user configuration file for remote workers.

```
# nickname for bastion host
Host work
       Hostname dmz.someplace.com
       Port 2929
       User max

# Defaults - must login via an identity key using only protocol 2.
Host *
       CheckHostIP yes
       Compression yes
       CompressionLevel 9
       ConnectionAttempts 3
       DSAAuthentication yes
       FallBackToRsh no
       ForwardAgent no
       ForwardX11 yes
       GatewayPorts no
       KeepAlive yes
       LocalForward 8080 intranet.extremefoosticks.com:80
       PasswordAuthentication no
       Protocol 2
       PubkeyAuthentication  yes
       RhostsAuthentication no
```

```
        RhostsRSAAuthentication no
        RSAAuthentication no
        StrictHostKeyChecking  yes
        UsePriviledgedPort no
        UseRsh no
        XAuthLocation /usr/X/bin/xauth
```

Workstation Configuration File

The following is an example of the Secure Shell user configuration file for a workstation.

```
# nickname for remote server
Host server
       HostName server.faroff.corp

# remote host needing a network proxy to access.
Host remote
       HostName remote.otherplace.org
       User pablo
       ProxyCommand /usr/lib/ssh/ssh-socks5-proxy-connect -h socks.server
-p 1080 remote.otherplace.org 22

# Defaults
Host *
       CheckHostIP yes
       Compression yes
       CompressionLevel 6
       FallBackToRsh no
       ForwardAgent no
       ForwardX11 yes
       GatewayPorts no
       KeepAlive yes
       PasswordAuthentication yes
       Protocol 2
       StrictHostKeyChecking ask
       UseRsh no
       XAuthLocation /usr/X/bin/xauth
```

About the Author

Jason Reid (jason.m.reid@sun.com) is in the Solaris System Test group. Prior to his present position, he was an SQA engineer in the Developer Tools Group. Before joining Sun, Jason worked at the Purdue University Computing Center as an UNIX systems administrator, while obtaining his B.S. in Computer Science.

References

Anderson, Ross. *Security Engineering A Guide to Building Dependable Distributed Systems*. New York: Wiley Computer Publishing, 2001.

Barrett, Daniel J. and Richard E. Silverman. *SSH: The Secure Shell*. Sebastopol, CA: O'Reilly, 2001.

Frisch, Aeleen. *Essential System Administration, Second Edition*. Sebastopol, CA: O'Reilly, 1995.

Garfinkel, Simson and Gene Spafford. *Practical Unix and Internet Security, Second Edition*. Sebastopol CA: O'Reilly, 1996.

OpenBSD Group, "OpenSSH Frequently Asked Questions," OpenBSD Group, April 2002, at:
http://www.openssh.com/faq.html

Reid, Jason and Keith Watson, "Building and Deploying OpenSSH for the Solaris Operating Environment," Sun BluePrints OnLine, July 2001, at:
http://www.sun.com/blueprints/0701/openSSH.pdf

Reid, Jason, "Building OpenSSH—Tools and Tradeoffs," Sun BluePrints OnLine, January 2003, at:
http://www.sun.com/blueprints/0103/817-1307.pdf

Reid, Jason, "Configuring OpenSSH for the Solaris Operating Environment," Sun BluePrints OnLine, January 2002, at:
http://www.sun.com/blueprints/0102/configssh.pdf

Ordering Sun Documents

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

Accessing Sun Documentation Online

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

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