HP-UX Secure Shell A.03.81 Release Notes

HP-UX 11.0, 11i v1, and 11i v2

Edition 1

Manufacturing Part Number : T1471-90011
July 2004

USA

Legal Notices

The information contained herein is subject to change without notice.

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Printed in the U.S.A.

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Trademark Notices. UNIX is a registered trademark of The Open Group.

Contents

HP-UX Secure Shell A.03.81

Overview . . . 8
New Features . . . 9
Use of Untrusted Cookies for X11-Forwarding . . . 9
Support for Sending Application Layer Keep-Alive Messages to the Server . . . 10
The /etc/moduli File Updated . . . 11
Support for GSSAPI Replaced with GSSAPI-With-MIC . . . 11
Unsupported Features . . . 14
Fixes in HP-UX Secure Shell A.03.81 . . . 15
HP-UX Secure Shell and the Strong Random Number Generator . . . 17
HP-UX Secure Shell Resources . . . 18
Getting HP-UX Secure Shell Software . . . 18
Getting More Information About Secure Shell Technology . . . 18
HP-UX Secure Shell Commands . . . 19
Installation Requirements . . . 20
Patch Requirements . . . 20
System Requirements . . . 21
Operating System . . . 21
Hardware . . . 21
Disk Space . . . 21
Software Availability in Native Languages . . . 21
Installing HP-UX Secure Shell . . . 22
Configuring HP-UX Secure Shell . . . 23
Password Authentication . . . 23
Key Generation . . . 24
Public Key Authentication . . . 24
Port Forwarding . . . 25
X11 Forwarding . . . 26
Configuration Directive Settings in the sshd_config File . . . 27
AllowGroups . . . 27
AllowTCPForwarding . . . 27
AuthorizedKeysFile . . . 28
ChallengeResponseAuthentication . . . 28
ClientAliveCountMax . . . 28
ClientAliveInterval . . . 29
Compression . . . 29

Contents

DenyGroups 30
DenyUsers 30
GatewayPorts 30
GSSAPIAuthentication 31
GSSAPICleanupCredentials 31
GSSAPIEnableMitmAttack 31
HostbasedAuthentication 32
HostKey 32
IgnoreRhosts 33
IgnoreUserKnownHosts 33
KerberosAuthentication 33
KerberosGetAFSToken 34
KerberosOrLocalPasswd 34
KerberosTicketCleanup 35
ListenAddress 35
LoginGraceTime 35
LogLevel 36
MACs 36
MaxStartups 36
PasswordAuthentication 37
PermitEmptyPasswords 37
PermitRootLogin 37
PermitUserEnvironment 38
PidFile 38
Port 39
PrintLastLog 39
PrintMotd 40
Protocol 40
PubkeyAuthentication 40
RhostsRSAAuthentication 41
RSAAuthentication 41
StrictModes 42
SyslogFacility 42
TCPKeepAlive 43
UseDNS 43
UseLogin 44

Contents

UsePAM 44
UsePrivilegeSeparation 45
X11DisplayOffset 45
X11Forwarding 45
X11UseLocalhost 46
HP-UX Secure Shell and chroot-ed environments 47
New user and chroot configuration 47
Configuring chroot environment for existing users 48
Configuring chroot for sftp 48
Configuring chroot for ssh 51

Contents

HP-UX Secure Shell A.03.81

This release note contains information for HP-UX Secure Shell A.03.81. It includes new features, known problems and workarounds, and details on the supported platforms. Use this information to get started using HP-UX Secure Shell A.03.81. This document does not include all configurations and situations (additional information resources are described later in this document).

NOTE

Go to `http://software.hp.com` for the most recently released HP-UX Secure Shell software.

Go to the "Internet and Security Solutions" page at `http://docs.hp.com` for the most recently released HP-UX Secure Shell documentation.

Overview

HP-UX Secure Shell A.03.81 is based on OpenSSH 3.8.1p1. The client/server architecture supports the SSH-1 and SSH-2 protocols and provides secured remote login, file transfer, and remote command execution. The product is available for HP-UX 11.0, 11i version 1, and 11i version 2.

HP-UX Secure Shell uses hashing to ensure data integrity and provides secure tunneling features, port forwarding, and an SSH agent to maintain private keys on the client.

HP-UX Secure Shell supports the following authentication methods:

- Kerberos 5/GSSAPI
- Password
- Public key
- Keyboard-Interactive
- Host-based

The following technologies are tested with HP-UX Secure Shell:

- Kerberos 5/GSSAPI
- OpenSSL
- IPv6
- Trusted Systems
- TCP Wrappers
- PAM (PAM_UNIX, PAM_Kerberos, PAM_LDAP)

HP-UX Secure Shell A.03.81 contains the following libraries:

- zlib v1.2.1
- OpenSSL v0.9.7d
- TCP Wrappers v7.6

HP-UX Secure Shell is a fully tested HP product. HP supports HP-UX Secure Shell at no additional cost to customers with HP-UX support agreements.

New Features

Following are the new features in HP-UX Secure Shell A.03.81:

- "Use of Untrusted Cookies for X11-Forwarding" on page 9
- "Support for Sending Application Layer Keep-Alive Messages to the Server" on page 10
- "The /etc/moduli File Updated" on page 11
- "Support for GSSAPI Replaced with GSSAPI-With-MIC" on page 11

The subsequent sections contain a detailed description of the new features.

Use of Untrusted Cookies for X11-Forwarding

This feature has been introduced as of OpenSSH 3.8.1p1 to counteract the SSH client xauth vulnerability as described in http://www.securityfocus.com/bid/1006/info/.

A vulnerability exists in the default configuration of the SSH client that could be used to compromise the security of a client machine. By default, ssh clients will negotiate to forward X connections. This is done using the `xauth` program to place cookies in the authorization cache of the remote machine for the user logging in. If the superuser on the remote host cannot be trusted, or the root account has been compromised, the xauth key can be read from the `.Xauthority` file of the user, and used to connect to the client machine. This can result in a wide range of compromises on the client host.

When an application needs X11 forwarding, it is expected to establish a connection with a local X-server. The common mode of authorization with the X-server is through an `.Xauthority` Cookie. At the beginning of the X-protocol session, the application submits this (trusted) cookie to the X-server to gain access to the remote host. To avoid potential deciphering of the cookie by another user on an unsecured network, starting with this release clients who need X11 forwarding will generate untrusted cookies by default. A client with an untrusted cookie will automatically be restricted from accessing other trusted clients and servers, thereby adding more security to X11 forwarding.

Users must explicitly restore trusted cookies by setting `ForwardX11Trusted = yes` in the `sshd_config` file. This was also the default setting in previous releases of Secure Shell. This is a default change between Secure Shell versions A.03.71.xxx and this new version.

IMPORTANT Starting with release A.03.81, by default, SSH assumes all clients to be untrusted (ForwardX11Trusted no), and denies them access to the target machine. To have full access, the ForwardX11Trusted yes setting has to be used (also the default in SSH 3.7).

Support for Sending Application Layer Keep-Alive Messages to the Server

Starting with this release, HP-UX Secure Shell supports a set of four configuration directives to control the duration for which clients stay connected to unresponsive servers, and the duration for which a server keeps the connection alive with unresponsive clients. Prior to this release, the ssh client depended on the TCP connection timeout value to close the connection with the server. With this release, the user has greater control over inactive connections.

`ServerAliveInterval` Sets the timeout interval (in seconds) after which, if no data has been received from the server, the client sends a message through the encrypted channel to request a response from the server. The default is 0, indicating that these messages will not be sent to the server. This option applies to protocol version 2 only.

`ServerAliveCountMax` Sets the number of server alive messages, which may be sent without receiving any messages back from the server. If this threshold is reached while the server alive messages are being sent, the client will disconnect from the server. This terminates the session.

Prior to this release, only the `sshd` server had the provision to send such keep alive messages to unresponsive clients.

`ClientAliveInterval` This directive is protocol version 2 specific. It is used to send a request to unresponsive clients.This directive sets a timeout interval in seconds after which, if no data has been received from the client, sshd sends a message through the encrypted channel to request a response from the client. The default is 0 (no message will be sent to the client). Also, see `ClientAliveCountMax`.

`ClientAliveCountMax` Sets the number of client alive messages, which may be sent even if `sshd` does receive any message back from the client. If this threshold is reached while client alive messages are being sent, sshd will disconnect the client. Thus, terminating the session. It is important to note that the use of client alive messages is very different from `KeepAlive`. The client alive messages are sent through the encrypted

channel and therefore will not be spoofable. The client alive mechanism is valuable when the client or server depend on knowing when a connection has become inactive.

The /etc/moduli File Updated

The `/etc/moduli` file is updated to improve the performance of SSH protocol version 2. For more information, please refer to `http://www.ietf.org/internet-drafts/draft-ietf-secsh-dh-group-exchange-04.txt` and `http://www.openbsd.org/cgi-bin/cvsweb/src/etc/Makefile?rev=HEAD`

Support for GSSAPI Replaced with GSSAPI-With-MIC

The experimental "gssapi" support has been replaced with the "gssapi-with-mic" to fix possible MITM attacks. The `gssapi-with-mic` utility is an improved security mechanism, done with the help of a message integrity code (MIC). Access is allowed only if the client sends a valid message integrity code (MIC) binding the GSSAPI context to the keys used for encryption/integrity protection of the SSH session. This prevents a man-in-the-middle attack that could eavesdrop on the authentication messages between the real client and server.

NOTE With this release, the previous GSSAPI authentication method has been deprecated. The two authentication methods are incompatible. By default, older versions of HP-UX Secure Shell clients will not be able to connect to an SSH 3.8 server using GSSAPI authentication, and also, by default, an SSH 3.8 client will not be able to do GSSAPI authentication against an older version of the SSH server.

However, HP-UX Secure Shell A.03.81 has been built with a patch that provides continued support for the deprecated GSSAPI-based authentication. With this patch, it is possible for older versions of SSH clients to connect to an SSH 3.8 server using GSSAPI authentication, and also for an SSH 3.8 client to connect to older versions of SSH servers using GSSAPI authentication.

For the patch to take effect, the user must explicitly set the `GssapiEnableMitmAttack` directive to `yes` ("GssapiEnableMitmAttack yes") in either the client or the server or both client and server, as appropriate. The `GssapiEnableMitmAttack` configuration directive is new with this release, and is required for both the client configuration (`ssh_config`) and server configuration (`sshd_config`).

This new configuration directive is not documented in the `ssh_config` or `sshd_config` man pages. The following examples should help illustrate the use of this new directive:

- If an SSH 3.7 client wants to connect to an SSH 3.8 server using GSSAPI authentication, the user must set the `GssapiEnableMitmAttack` directive to `yes` in the server configuration file. This enables the patch on the server, and provides support for the deprecated GSSAPI authentication method.
- If an SSH 3.8 client wants to connect to an SSH 3.6 server using GSSAPI authentication, the user must set the `GssapiEnableMitmAttack` directive to `yes` in the client configuration file. This allows the client to explicitly tell the server that it wants the (deprecated) GSSAPI authentication method.
- If an SSH 3.8 client wants to connect to an SSH 3.8 server, and wants to use the (deprecated) GSSAPI authentication method, then the user must set the `GssapiEnableMitmAttack` directive to `yes` for both the client and server configurations. This enables the patch on the server side, and it forces the client to change its preferred authentication to be the (deprecated) GSSAPI method.
- If an SSH 3.7 client wants to connect to an SSH 3.8 server using GSSAPI authentication, and the user has not set the `GssapiEnableMitmAttack` directive to `yes` in the server configuration file, the connection attempt will terminate in error. This is because the SSH 3.7 client will be asking for the deprecated GSSAPI authentication method, and the server will not have that method enabled.
- If an SSH 3.8 client wants to connect to an SSH 3.6 server using GSSAPI authentication, and the user has not set the `GssapiEnableMitmAttack` directive to `yes` in the client configuration file, the connection attempt will terminate in error. This is because the client's authentication method will be set to `GssapiEnableMitmAttack` by default, and the older server (SSH 3.6) will not recognize that method.
- If an SSH 3.8 client wants to connect to an SSH 3.8 server, and wants to use the (deprecated) GSSAPI authentication method, and if the user sets the `GssapiEnableMitmAttack` directive to `yes` on the client, but not on the server, the connection will terminate in error. This is because client will be asking for the (deprecated) GSSAPI method, and the server will not have been enabled to support it.
- If an SSH 3.8 client wants to connect to an SSH 3.8 server using GSSAPI authentication, and if the user sets the `GssapiEnableMitmAttack` directive to `yes` on the server, but not on the client, the connection will succeed, but will use the new `gss-with-mic` authentication method. This is because although the server will be enabled to support the (deprecated) GSSAPI method, the client will be asking for the default GSSAPI authentication method, which, for the SSH 3.8 client is the `gss-with-mic` authentication method.
- If an SSH client (any version) wants to connect to an SSH server (any version) using an authentication method other than GSSAPI, then the `GssapiEnableMitmAttack` directive is not relevant.

Although the deprecated GSSAPI method continues to be supported in this release of HP-UX Secure Shell, HP recommends that new installations use the gssapi-with-mic mechanism. Existing installations of (older versions of) HP-UX Secure Shell are encouraged to upgrade to this new version at the earliest opportunity.

Unsupported Features

The following features are unsupported beginning with Secure Shell A.03.81:

- `KerberosGetAFSToken` option for sshd(8)

  This directive is used to specify whether or not to accept forwarded Andrew file system (AFS) tokens. HP-UX Secure Shell A.03.81 does not support this feature.

- Host keys in DNS (draft-ietf-secsh-dns-xx.txt)

  This is a configuration option for SSH. HP-UX Secure Shell A.03.81 does not support this feature. Refer to the README.dns for more information.

Fixes in HP-UX Secure Shell A.03.81

HP-UX Secure Shell A.03.81 is based on OpenSSH 3.8.1p1 and contains several fixes. Table 1 contains a list of defects fixed in this version of HP-UX Secure Shell.

Table 1 Fixes in HP-UX Secure Shell A.03.81

Identifier	Description
JAGaf18995	Shadow password causes SSH to fail for an LDAP user.
JAGaf07528	`LoginGraceTime` in SSH does not work as expected.
JAGaf16650	The bad login records written to the `btmp` file should show client host name, and not just client IP address.
JAGaf16658	An option needs to be provided for `scp` not to print the banner, in SSH.
JAGaf20120 JAGaf20125	Credentials are not stored on the server after successful authentication of `PAM_KERBEROS` or `PAM_DCE`.
JAGaf24109 JAGaf18752	The `password` authentication method should not ignore the `UsePAM` setting.
JAGaf10129 JAGaf16892	SSH does not sync the `utmpx` and `wtmpx` files on HP-UX 11i v2 systems. This defect was also fixed for SSH A.03.71.002.
JAGaf20208	Secure Shell needs to be rebuilt with the latest version of `zlib`. **NOTE:** zlib version 1.2.1 is now included with HP-UX Secure Shell A.03.81.
JAGaf26760	The `chroot` documentation has been fixed to include a set of new files that need to be copied to properly support UsePAM YES, and also to properly support some shell commands from within sftp
JAGaf24109	The password authentication method should not ignore the `UsePAM` setting.

Refer to http://bugzilla.mindrot.org, for more information. In addition to the defect fixes in Table 1, the following fixes are available in this version of HP-UX Secure Shell:

- Some memory leaks have been fixed.

HP-UX Secure Shell and the Strong Random Number Generator

HP-UX Secure Shell requires that a random number generator be present on the system. It looks for `/dev/urandom` and `/dev/random` (in that sequence) and uses the first device it finds. If it fails to locate these two devices, it uses its own internal random number generator program. The /dev/urandom and `/dev/random` devices are available by default on HP-UX 11i v2 systems. The `/dev/random` device is also available for, but not installed by default, on HP-UX 11i v1 systems. HP recommends that Secure Shell users on HP-UX 11i v1 systems install the Strong Random Number Generator product, as it significantly speeds up program initialization and execution time for some commands.

For more information on the Strong Random Number Generator product, please refer to http://www.software.hp.com.

HP-UX Secure Shell Resources

Getting HP-UX Secure Shell Software

You can get HP-UX Secure Shell from the following sources:

- HP Software Depot at `http://software.hp.com`
- HP-UX Application Release CDs
- HP-UX 11i version 2 Operating Environment (OE)

The most recent version of HP-UX Secure Shell is always available at `http://software.hp.com`. HP also provides HP-UX Secure Shell on Application Release CDs, however, the CDs may not always offer the most recent version of HP-UX Secure Shell.

Getting More Information About Secure Shell Technology

A large volume of information exists for Secure Shell technology. HP recommends learning more by reading O'Reilly's *SSH, The Secure Shell - The Definitive Guide* by Daniel J. Barrett and Richard E. Silverman.

This HP-UX Secure Shell Release Notes document is available at the following locations:

- HTML and pdf versions at `http://docs.hp.com` (*Internet and Security Solutions*)
- A Readme text version in the software at: `/opt/ssh/README.hp`
- The HP Instant Information CD starting with Application Release 0902

You can also learn about Secure Shell technology at the following locations:

- OpenSSH at `http://www.openssh.com`
  - — FAQs, Mail List Archives, Security pages, manpages
- IETF at `http://www.ietf.org/` (go to Working Groups > Security)
- The HP book *HP-UX 11i Security* by Chris Wong. A portion of the HP-UX Secure Shell content is available at `http://searchsystemsmanagement.techtarget.com` on the "Tips and Newsletters" page in the "HP-UX SysAdmin" section.
- SSH Secure Shell FAQs at:
  `http://www.employees.org/~satch/ssh/faq/ssh-faq.html`

HP-UX Secure Shell Commands

The following table lists the HP-UX Secure Shell commands and provides a brief description of each. Refer to the manpage for each command for more information.

Table 2 HP-UX Secure Shell Commands

Command	Description
`ssh`	client program similar to `rlogin` and `rsh`
`sshd`	secure shell server daemon
`sftp`	secure `ftp` program
`scp`	secure file copy program similar to `rcp`
`slogin`	symbolic link to `ssh`
`ssh-agent`	authentication agent to store private keys
`ssh-add`	tool for adding keys to `ssh-agent`
`ssh-keygen`	tool for manually creating public and private keys
`sftp-server`	`sftp` server subsystem automatically initiated by `sshd`
`ssh-keyscan`	tool for gathering public host keys

Installation Requirements

Patch Requirements

HP has tested HP-UX Secure Shell A.03.81 in test environments with the following Support Plus patch bundles installed. While HP recommends HP-UX 11.0 customers install the listed patch bundle, HP mandates HP-UX 11i version 1 customers install the listed patch bundle.

Table 3 Quality Packs Tested and Required for HP-UX 11.00 and 11.11

OS	Recommended Support Plus Patch Bundle Date / Release # / Part #
HP-UX 11.00	March 2003 SP60 Quality Pack
HP-UX 11i version 1.0	December 2002 Fusion 1202 Pack

More information about Support Plus patch bundles is available on the IT Resource Center at: `http://itrc.hp.com`.

HP recommends installing the following `libc` patches for use with HP-UX Secure Shell A.03.81:

Table 4 libc Patch Bundle Recommended

OS	Patches Tested	Recommended Patch Bundle
HP-UX 11.0	PHCO_25976	PHCO_25976
HP-UX 11i version 1	PHCO_27740	PHCO_27740

PAM patches: HP recommends installing the following PAM patches for use with HP-UX Secure Shell A.03.81:

Table 5 PAM Patches - Tested and Recommended

OS	Patches Tested	Patches Recommended
HP-UX 11.00	PHCO_26089,PHCO_29249	PHCO_29249
HP-UX 11i version 1.0	PHCO_27064,PHCO_30402	PHCO_30402

pthreads patches: HP recommends installing the following pthreads patches for use with HP-UX Secure Shell A.03.81:

Table 6 pthreads Patches - Tested and Recommended

OS	Patches Tested	Patches Recommended
HP-UX 11.00	PHCO_26960	PHCO_26960
HP-UX 11i version 1.0	PHCO_26466	PHCO_26466

System Requirements

The following are the minimum requirements for HP-UX Secure Shell:

NOTE HP tested HP-UX Secure Shell A.03.81 with OpenSSH 3.8.1p1 and expects compatibility with previous versions of OpenSSH.

Operating System

- HP-UX 11.0
- HP-UX 11i version 1
- HP-UX 11i version 2

Hardware

HP/9000 Servers and Workstations

HP Itanium® Servers and Workstations

Disk Space

HP-UX Secure Shell A.03.81 requires approximately 32 MB on all supported OS versions and platforms.

Software Availability in Native Languages

English only

Installing HP-UX Secure Shell

You do not need to remove previous versions of HP-UX Secure Shell before upgrading to a newer version. However, if you are downgrading to an older version of HP-UX Secure Shell, HP recommends removing the product before installing the older version.

Follow the steps below to use `swinstall` and install HP-UX Secure Shell:

Step 1. Log in as root.

Step 2. Insert the software CD into the appropriate drive if installing from the Application Release CD. If installing from http://software.hp.com, download the depot and use the `swinstall` directions provided on the Installation page where you downloaded the software.

Step 3. Run `$ swinstall -s <fully-qualified depot source path>`

Step 4. Enter the drive mount point in the `Source Depot Path` field and click **OK**. Change the `Source Host Name` if needed.

Step 5. Select `T1471AA` from the list of available software and click `Mark for Install` on the `Actions` menu.

Step 6. Click `Install` on the `Actions` menu to begin installation.

Step 7. Click **OK** in the `Install Analysis` window when the `Status` field displays a Ready message.

Step 8. Click **Yes** to begin the installation. `swinstall` loads the HP-UX Secure Shell files in approximately 3 to 5 minutes.

NOTE The `sshd` daemon is preconfigured and starts after installation.

The `swinstall` command installs HP-UX Secure Shell in the `/opt/ssh/` directory.

Configuring HP-UX Secure Shell

Use the following information as a supplement to the manpages and O'Reilly's *SSH, The Secure Shell -- The Definitive Guide*.

HP recommends using SSH-2 to eliminate the risk of an insertion attack. The installation script generates RSA1, RSA, and DSA server keys if these did not previously exist. The following is a list of the host keys generated and placed in `/opt/ssh/etc/` upon installation:

Table 7 Host Keys HP-UX Secure Shell Generates Upon Installation

Description	Host Key
RSA1 private and public host keys for SSH-1	`ssh_host_key` `ssh_host_key.pub`
RSA private and public host keys for SSH-2	`ssh_host_rsa_key` `ssh_host_rsa_key.pub`
DSA private and public host keys for SSH-2	`ssh_host_dsa_key` `ssh_host_dsa_key.pub`

You will find the HP-UX Secure Shell configuration files at the following locations:

Table 8 Configuration Files

File	Location
server configuration file	`/opt/ssh/etc/sshd_config`
client configuration file	`/opt/ssh/etc/ssh_config`

Password Authentication

HP-UX Secure Shell uses PAM for password authentication. `sshd` uses its own configuration lines in `/etc/pam.conf`. You do not need to edit `/etc/pam.conf` for `sshd` to use it because of the OTHER service in `pam.conf`. Unspecified applications in `/etc/pam.conf` always use the OTHER service. You can create a link to `sshd` and change the service name in `/etc/pam.conf` to the link name. However, the service name in `/etc/pam.conf` must match the name of the daemon invoked. For example, if you create a link to `sshd` named `lsshd` and then invoke `lsshd`, PAM looks for the service name `lsshd`. For more information on configuring PAM, refer to the `pam.conf(4)` manpage.

HP-UX Secure Shell supports the PAM interface. Starting with OpenSSH Version 3.7, a new configuration directive, `UsePAM`, has been introduced in the server configuration file, `sshd_config` file, to enable or disable the PAM support. The `UsePAM` directive is enabled by default in the configuration file.

Following is a sample usage of the `UsePAM` directive:

```
UsePAM yes
UsePAM no
```

In the previous releases of OpenSSH, PAM support was enabled or disabled using the `#define USE_PAM 1` definition in the `config.h` header file.

HP-UX Secure Shell has been tested with HP PAM implementations (PAM_UNIX, PAM_Kerberos, PAM_LDAP) and HP-UX Secure Shell works with any other well-written PAM module.

Key Generation

By default, HP-UX Secure Shell is set to use SSH-2 only. The default setting for `ssh-keygen` is null. You must specify which type of keys you want `ssh-keygen` to generate. `ssh-keygen` can create SSH-1 (RSA1) and SSH-2 (RSA, DSA) key pairs. Use the `-t` option to generate SSH-2 key pairs. For example, `ssh-keygen -t dsa` generates SSH-2 DSA key pairs. Refer to the `ssh-keygen` manpage for more information.

Public Key Authentication

Generate key pairs for the user at the client system. Following are the acceptable values for the `t` option:

- RSA
- DSA
- RSA1

Use RSA1 to generate keys for the SSH-1 protocol, and use RSA and DSA to generate keys for the SSH-2 protocol. Use the default file name `~/.ssh/identity` for SSH-1, or `~/.ssh/id_rsa` and `~/.ssh/id_dsa` for SSH-2.

NOTE The RSA1 and RSA value will generate keys in different length.

The following are examples of `ssh-keygen` commands:

```
#ssh-client-system ssh-keygen -t rsa1
#ssh-client-system ssh-keygen -t rsa
#ssh-client-system ssh-keygen -t dsa
```

Depending on which protocol and algorithm you use, securely copy or append the public key file (`~/.ssh/identity.pub`, `~/.ssh/id_dsa.pub`, or `~/.ssh/id_rsa.pub`) from the ssh-client system to the ssh-server system and name it `~/.ssh/authorized_keys`. You can append the file using the `cat` command.

Port Forwarding

Port forwarding typically consists of using a local port in place of a remote port. How you invoke `ssh` depends on whether you want port forwarding to be persistent or not.

1. Let `<lport>` be a port on your local system that you have access to and greater than 1024 if you are not root. Be sure the port is not being used.
2. Let `<rmachine>` be the remote system you wish to connect to.
3. Let `<rport>` be the remote port on `<rmachine>`.
4. The `-L` option specifies port mapping.
5. `<command>` runs on the remote machine and should not have any terminal interaction.

   You can use `sleep` as the `<command>`. Use `sleep` with a large number to make the connection persistent (`sleep` with a parameter of 1000000 will last over 11 days). If you use `sleep` with a small number like 10, you can use port forwarding for only 10 seconds. However, if the port being forwarded is in use, the port will not close until the connection terminates. Therefore, if you use a short sleep period and use port forwarding longer than the sleep period, the connection will last as long as it is used and then stop.
6. Connect to `<lport>` on the localhost to use the port forwarding connection, as follows:

   ```
   $ ssh -L<lport>:localhost:<rport> <rmachine> <command>
   ```

For example: You want to connect to a service on remote machine B, which listens on port 123. Your application resides locally on machine A and is configured to connect to port 123 on machine B. Start a Secure Shell port forwarding session with the following:

```
ssh -L22222:localhost:123 machineB sleep 1000000
```

This configures your application to connect to port 22222 on localhost. The application opens a connection to port 22222 on localhost, where `ssh` waits for a connection. `ssh` sends all communication to `sshd` on machine B, which opens a connection to port 123. All communication travels as if there was a direct connection, however communication will be through the SSH encrypted connection.

IMPORTANT When using port forwarding to an IPv6-enabled machine running HP-UX Secure Shell, the `GatewayPorts` option in `sshd_config` command must be set to `yes` on the IPv6-enabled HP-UX machine. Not setting the `GatewayPorts` option to `yes` will cause applications to fail to connect to the specified port.

X11 Forwarding

You can modify `sshd_config` and `ssh_config` to allow X11 forwarding. Edit `sshd_config` to set the display instance (default = 10). After establishing an SSH connection to a remote system, HP-UX Secure Shell will have set the DISPLAY environment variable to the following:

```
<server machine>:<display instance>.0
```

`<server machine>` is the machine you connected to when you initiated the Secure Shell connection. SSH attempts to set `<display instance>` to the display instance specified in the configuration file. However, if that port is in use, SSH sets it to a different number that is greater than the display instance specified. X programs displays on the originating machine when started.

IMPORTANT When using X11 forwarding to an IPv6-enabled machine running HP-UX Secure Shell, the `X11UseLocalhost` option may not work. If the machine has the TOUR (Transport Optional Upgrade Release) Version 1.0 or greater installed, `X11UseLocalhost` can be set to either `yes` or `no`. If the TOUR is not installed, the `X11UseLocalhost` option must be set to `no` in `sshd_config` on the IPv6-enabled machine. X11 applications will not connect to the display if the `X11UseLocalhost` option is not set to `no`.

IMPORTANT Set the `X11UseLocalhost` value to `no` if the X11 client uses a pre-X11R6 library for X11 forwarding.

Configuration Directive Settings in the sshd_config File

The HP-UX Secure Shell daemon (sshd) reads a set of runtime configuration directives from the `sshd_config` file. By default, this file is located in the `/opt/ssh/etc` directory. These configuration directives control `sshd` behavior for various functions. Following is a comprehensive list of valid configuration flags along with their default values, as set by HP-UX Secure Shell.

NOTE A * in front a configuration directive value implies that this default on HP-UX has been modified from its original OpenSSH default value.

AllowGroups

This keyword can be followed by a list of group name patterns, separated by spaces. If specified, login is allowed only for users whose primary group or supplementary group list matches one of the patterns. '*' and '?' can be used as wildcards in the patterns. Only group names are valid; a numerical group ID is not recognized. By default, login is allowed for all groups. The default condition is when this directive is not specified in the sshd_config file. This directive has been available for a last several releases.

AllowTCPForwarding

This directive is used to specify whether TCP forwarding is permitted. The default is `yes`..Disabling TCP forwarding does not improve security unless users are also denied shell access, as they can always install their own forwarders.

Table 9 AllowTCPForwarding Default Values

SSH Version	Default Values
SSH 3.6	NA
SSH 3.7	YES
SSH 3.8	YES

AuthorizedKeysFile

This directive specifies the filename that can be used for public key authentication. The `AuthorizedKeysFile` may contain tokens of the form `%T` which are substituted during connection setup.The following tokens are defined:

- `%%` is replaced by a `%`
- `%h` is replaced by the home directory of the user being authenticated
- `%u` is replaced by the username of that user

After expansion, `AuthorizedKeysFile` is taken to be an absolute path or one relative to the home directory of the user.

Table 10 AuthorizedKeysFile Default Values

SSH Version	Default Values
SSH 3.6	`.ssh/authorized_keys`
SSH 3.7	`.ssh/authorized_keys`
SSH 3.8	`.ssh/authorized_keys`

ChallengeResponseAuthentication

This directive is used to enable `ChallengeResponseAuthentication`. This controls the use of PAM for authentication specifically in `FreeBSD`. This affects the effectiveness of the `PasswordAuthentication` and `PermitRootLogin` variables.

Table 11 ChallengeResponseAuthentication Default Values

SSH Version	Default Values
SSH 3.6	NA
SSH 3.7	YES
SSH 3.8	YES

ClientAliveCountMax

This directive sets the number of client alive messages, which may be sent without `sshd` receiving any messages back from the client. If this threshold is reached while client alive messages are being sent, `sshd` will disconnect the client. Thus, terminating the session.It is

important to note that the use of client alive messages is different from `KeepAlive`. The client alive messages are sent through the encrypted channel and therefore will not be spoofable.The client alive mechanism is valuable when the client or server depend on knowing when a connection has become inactive.

Table 12 ClientAliveCountMax Default Values

SSH Version	Default Values
SSH 3.6	NA
SSH 3.7	3
SSH 3.8	3

ClientAliveInterval

This directive is protocol version 2 specific and is used to send a request to nonresponsive clients.This directive sets the timeout interval in seconds after which if no data has been received from the client, `sshd` sends a message through the encrypted channel to request a response from the client.

Table 13 ClientAliveInterval Default Values

SSH Version	Default Values
SSH 3.6	NA
SSH 3.7	0
SSH 3.8	0

Compression

This directive is used to compress SSH connections.This directive when enabled, compresses the data sent over the SSH connection before it is encrypted. It also compresses data received on the client side in uncompressed form after being decrypted at the client side.

Table 14 Compression Default Values

SSH Version	Default Values
SSH 3.6	YES
SSH 3.7	YES

Table 14 Compression Default Values (Continued)

SSH Version	Default Values
SSH 3.8	YES

DenyGroups

This keyword can be followed by a list of group name patterns, separated by spaces. Login is disallowed for users whose primary group or supplementary group list matches one of the patterns. '*' and '?' can be used as wildcards in the patterns. Only group names are valid; a numerical group ID is not recognized. By default, login is allowed for all groups.

DenyUsers

This keyword can be followed by a list of user name patterns, separated by spaces. Login is disallowed for user names that match one of the patterns. '*' and '?' can be used as wildcards in the patterns. Only user names are valid; a numerical user ID is not recognized. By default, login is allowed for all users. If the pattern takes the form `USER@HOST` then `USER` and `HOST` are separately checked, restricting logins to particular users from particular hosts.

GatewayPorts

This directive is used to permit any host to connect to locally forwarded ports. GatewayPorts can be used to specify that `sshd` should bind remote port forwardings to the wildcard address. Thus, allowing remote hosts to connect to forwarded ports. Disabling GetawayPorts make `sshd` bind remote port forwardings to the loopback address.This prevents other remote hosts from connecting to forwarded ports.

Table 15 GatewayPorts Default Values

SSH Version	Default Values
SSH 3.6	NO
SSH 3.7	NO
SSH 3.8	NO

GSSAPIAuthentication

Specifies whether the user authentication based on GSSAPI is allowed.

Table 16 GSSAPIAuthenticaion Default Values

SSH Version	Default Values
SSH 3.6	Support is given with a external patch
SSH 3.7	NO
SSH 3.8	NO

GSSAPICleanupCredentials

Specifies whether to automatically destroy the user's credentials cache on logout.

Table 17 GSSAPICleanupCredentials Default Values

SSH Version	Default Values
SSH 3.6	NA
SSH 3.7	YES
SSH 3.8	YES

GSSAPIEnableMitmAttack

Specifies whether the (deprecated) GSSAPI authentication method is to be enabled for the server. The default is NO. If set to YES, older versions of SSH clients (3.7, 3.6 and so on) will be able to connect to an SSH 3.8 server using GSSAPI authentication. This directive is new for SSH 3.8, and applies to Protocol 2 only.

This directive has been introduced specifically for the reason that with this release of HP-UX Secure Shell, the previous GSSAPI authentication method has been replaced by the new GSSAPI_WITH_MIC authentication method. Since the two authentication methods are not compatible (i.e. an SSH client requesting the older GSSAPI authentication method will not work with an SSH server that supports ONLY the new GSSAPI-WITH-MIC method), this

release of Secure Shell has been built with a patch that enables support for the older GSSAPI authentication method. Users must set the GSSAPIEnableMitmAttack directive to YES if they want the server to enable support for the older GSSAPI authentication method.

Table 18 GSSAPIEnableMitmAttack Default Values

SSH Version	Default Values
SSH 3.6	NA
SSH 3.7	NA
SSH 3.8	NO

HostbasedAuthentication

This directive is similar to `RhostsRSAAuthentication` but it applies to protocol version 2.This is used to specify whether `rhosts` or `/etc/hosts.equiv` authentication together with successful public key client host authentication is allowed. It checks `/etc/ssh_known_hosts` and `$HOME/.ssh/known_hosts` for the public key.

Table 19 Hostbased Authenticaion Default Values

SSH Version	Default Values
SSH 3.6	NO
SSH 3.7	NO
SSH 3.8	NO

HostKey

Specifies a file containing a private host key used by SSH. The default is /opt/ssh/etc/ssh_host_key for protocol version 1, and /opt/ssh/etc/ssh_host_rsa_key and /opt/ssh/etc/ssh_host_dsa_key for protocol version 2.

NOTE sshd will refuse to use a file if it is group/world-accessible. It is possible to have multiple host key files. “rsa1” keys are used for version 1 and “dsa” or “rsa” are used for version 2 of the SSH protocol.

IgnoreRhosts

This directive is used to specify the usage of `.rhosts` and `.shosts` file in `RhostsRSAAuthentication` and `HostbasedAuthentication`. This directive when enabled ignores those files but `/etc/hosts.equiv` and `/opt/ssh/etc/shosts.equiv` are still used.

Table 20 IgnoreRhosts Default Values

SSH Version	Default Values
SSH 3.6	YES
SSH 3.7	YES
SSH 3.8	YES

IgnoreUserKnownHosts

This directive is used to specify the usage of `$HOME/.ssh/known_hosts` file in `RhostsRSAAuthentication` and `HostbasedAuthentication`.This directive when enabled ignores those files.An entry is added in this file whenever an unknown host is connected.

Table 21 IgnoreUserKnownHosts Default Values

SSH Version	Default Values
SSH 3.6	NO
SSH 3.7	NO
SSH 3.8	NO

KerberosAuthentication

This directive is supported only if it is enabled at compile time with the option, `--with-kerberos`. Connections may be authenticated by using Kerberos ticket or password authenticated by server if PasswordAuthentication is enabled. This is done with the help of KDC. To use this option, the server needs a Kerberos servtab, which allows the verification of the KDC's identity.

Table 22 KerberosAuthentication Default Values

SSH Version	Default Values
SSH 3.6	YES*

Table 22 KerberosAuthentication Default Values (Continued)

SSH Version	Default Values
SSH 3.7	YES*
SSH 3.8	YES*

KerberosGetAFSToken

AFS token passing causes OpenSSH to establish Kerberos/AFS credentials on the remote host, based on existing credentials on the client. If AFS is active and the user has a Kerberos 5 TGT, an attempt will be made to acquire an AFS token before accessing the user's home directory.

Table 23 KerberosGetAFSToken Default Values

SSH Version	Default Values
SSH 3.6	NA
SSH 3.7	NA
SSH 3.8	NO

KerberosOrLocalPasswd

This directive is to used to specify validation of password by other mechanisms such as `/etc/passwd/`, if Password Authentication through Kerberos fails. This is useful in an environment where Kerberos is in use, but not by everyone.

Table 24 KerberosOrLocalPasswd Default Values

SSH Version	Default Values
SSH 3.6	YES
SSH 3.7	YES
SSH 3.8	YES

KerberosTicketCleanup

This directive is used to specify whether destroying of users ticket cache file is allowed.This directive when enabled automatically destroys the users cache file during logout.

Table 25 KerberosTicketCleanup Default Values

SSH Version	Default Values
SSH 3.6	YES
SSH 3.7	YES
SSH 3.8	YES

ListenAddress

Specifies the local addresses sshd should listen on. The following forms may be used:

- ListenAddress host|IPv4_addr|IPv6_addr
- ListenAddress host|IPv4_addr:port
- ListenAddress [host|IPv6_addr]:port

If port is not specified, sshd will listen on the address and all prior Port options specified. The default is to listen on all local addresses. Multiple ListenAddress options are permitted. Additionally, any Port options must precede this option for non port qualified addresses.

LoginGraceTime

This directive is used to make the server daemon wait for a specified period of time for the user to login successfully. The server will be disconnected after this period of time has elapsed. If the time period is 0,then there is no time limit for the user to login. Thereby, making the server to wait till the user logs in successfully or depending on other configurations. The unit of time is set in seconds.

Table 26 LoginGraceTime Default Values

SSH Version	Default Values
SSH 3.6	120
SSH 3.7	120
SSH 3.8	120

LogLevel

Gives the verbosity level that is used when logging messages from sshd. The possible values are:

- QUIET
- FATAL
- ERROR
- INFO
- VERBOSE
- DEBUG
- DEBUG1
- DEBUG2
- DEBUG3

The default is INFO. DEBUG and DEBUG1 are equivalent. DEBUG2 and DEBUG3 each specify higher levels of debugging output. Logging with a DEBUG level violates the privacy of users and is not recommended.

MACs

Specifies the available MAC (message authentication code) algorithms. The MAC algorithm is used in protocol version 2 for data integrity protection. Multiple algorithms must be comma-separated. The default is
`hmac-md5,hmac-sha1,hmac-ripemd160,hmac-sha1-96,hmac-md5-96`.

MaxStartups

This directive is used to conserve resources in the server machine.This directive specifies the maximum number of unauthenticated concurrent connections that can be allowed to `sshd`. Additional connections will be dropped until authentication succeeds or the `LoginGraceTime` expires for a connection.

The random early drop can be enabled by specifying the three colon separated values `start:rate:full` (10:30:60). The SSH daemon, `sshd`, will refuse connection attempts with a probability of `rate/100`' (30%) if there are currently `start` (10) unauthenticated connections. The probability increases linearly and all connection attempts are refused if the number of unauthenticated connections reaches `full` (60).

Table 27 MaxStartups Default Values

SSH Version	Default Values
SSH 3.6	10
SSH 3.7	10
SSH 3.8	10

PasswordAuthentication

This directive is used to accept password as a login proof identity. The login password is sufficient for this case. However, there is an exception to accept Kerberos Server password when `KerberosAuthentication` is enabled.

Table 28 PasswordAuthentication Default Values

SSH Version	Default Values
SSH 3.6	YES
SSH 3.7	YES
SSH 3.8	YES

PermitEmptyPasswords

When password authentication is allowed, it specifies whether the server allows login to accounts with empty password strings. The default is `no`.

PermitRootLogin

This directive is used to specify whether any user can login as root using `ssh`. You can use this directive in the following methods:

1. Disable all root logins
2. Enable all root logins with any authentication method

3. Enable root logins with limited authentication methods

Table 29 PermitRootLogin Default Values

SSH Version	Default Values
SSH 3.6	YES
SSH 3.7	YES
SSH 3.8	YES

PermitUserEnvironment

This directive is used to control environment processing.This directive specifies whether `~/.ssh/environment` and `environment= options` in `~/.ssh/authorized_keys` are processed by `sshd`. The `~/.ssh/environment` file should be writable only by the user; it need not be readable by anyone else. Enabling environment processing may enable users to bypass access restrictions. This option is automatically disabled when the `UseLogin` option is enabled.

Table 30 PermitUserEnvironment Default Values

SSH Version	Default Values
SSH 3.6	NO
SSH 3.7	NO
SSH 3.8	NO

PidFile

This directive specifies where to look for the process id of the daemon.This file contains the latest instance of the running `sshd`, if multiple daemons are running. This file will be empty if there is no listening daemon. This option is ineffective when the server daemon is started in the debug mode.

Table 31 PidFile Default Values

SSH Version	Default Values
SSH 3.6	NA
SSH 3.7	`/var/run/sshd.pid`

Table 31 PidFile Default Values (Continued)

SSH Version	Default Values
SSH 3.8	/var/run/sshd.pid

Port

This directive is used to make ssh daemon to listen to a particular port. The SSH client should also connect to same port to which the daemon is listening.

Table 32 Port Default Values

SSH Version	Default Values
SSH 3.6	22
SSH 3.7	22
SSH 3.8	22

PrintLastLog

This directive is used to display information at the login time. This directive is similar to PrintMotd. This will display date and time of the users last logged in.

Table 33 PrintLastLog Default Values

SSH Version	Default Values
SSH 3.6	YES
SSH 3.7	YES
SSH 3.8	YES

PrintMotd

This directive is used to display information at the login time. This is similar to `PrintLastLog` but it takes information from the `/etc/motd` file, if this directive is enabled.

Table 34 PrintMotd Default Values

SSH Version	Default Values
SSH 3.6	YES
SSH 3.7	YES
SSH 3.8	YES

Protocol

This directive is used to specify the version number of the protocol, which `sshd` supports. Multiple versions must be comma-separated. This directive specifies the list of versions for the client to select and not the priority in protocols. Specifying 2,1 is identical to 1,2.

Table 35 Protocol Default Values

SSH Version	Default Values
SSH 3.6	2
SSH 3.7	2
SSH 3.8	2

PubkeyAuthentication

This directive is used to specify whether `PublickeyAuthentication` is allowed. Here the users identity is verified by way of cryptographic keys. In case of protocol version 1, if this directive is enabled and if `RSAAuthentication no`, then the next authentication method is looked up. But in case of protocol 2 the `DSA` authentication method is used.

Table 36 PubkeyAuthentication Default Values

SSH Version	Default Values
SSH 3.6	YES
SSH 3.7	YES

Table 36 PubkeyAuthentication Default Values (Continued)

SSH Version	Default Values
SSH 3.8	YES

RhostsRSAAuthentication

This directive is used to perform RSA-based host Authentication in addition to normal `.rhosts` or `/etc/hosts.equiv` authentication. This directive will not work for outbound connections from privileged ports. This option applies to protocol version 1 only.

Table 37 RhostsRSAAuthentication Default Values

SSH Version	Default Values
SSH 3.6	NO
SSH 3.7	NO
SSH 3.8	NO

RSAAuthentication

This directive is used to specify whether `RSAAuthentication` is allowed. This looks for keys in `~/.ssh/identity`, `~/.ssh/identity.pub` for protocol version1 and `~/.ssh/id_rsa`, `~/.ssh/id_rsa.pub` for protocol version 2.

Table 38 RSAAuthentication Default Values

SSH Version	Default Values
SSH 3.6	YES
SSH 3.7	YES
SSH 3.8	YES

StrictModes

This directive is used for checking the access rights and permissions for the files. This directive when enabled make `sshd` check file modes and ownership of the user's files and home directory before accepting login. This is normally desirable because novices sometimes accidentally leave their directory or files world-writable.

Table 39 StrictModes Default Values

SSH Version	Default Values
SSH 3.6	YES
SSH 3.7	YES
SSH 3.8	YES

SyslogFacility

Gives the facility code that is used when logging messages from sshd. Following are the possible values are:

- DAEMON
- USER
- AUTH
- LOCAL0
- LOCAL1
- LOCAL2
- LOCAL3
- LOCAL4
- LOCAL5
- LOCAL6
- LOCAL7

The default is AUTH.

TCPKeepAlive

This directive is used to control the flow of TCP `KeepAlive` messages to other side. If they are sent, death of the connection or crash of one of the machines will be noticed. This avoids infinitely hanging sessions.

However, this means that connections will die if the route is down temporarily. On the other hand, if Keepalives are not sent, sessions may hang indefinitely on the server, leaving `ghost` users and consuming server resources. This one is different from `ClientAliveCountMax`. The TCP Keepalive option enabled by KeepAlive is spoofable.

Table 40 TCPKeepAlive Default Values

SSH Version	Default Values
SSH 3.6	YES
SSH 3.7	YES
SSH 3.8	YES

UseDNS

This directive is called `VerifyReverseMapping` (default no) in previous versions. This directive is specified in order for `sshd` to lookup the remote hostname and check that the resolved hostname for the remote IP address maps back to very same IP address.

Table 41 UseDNS Default Values

SSH Version	Default Values
SSH 3.6	NA
SSH 3.7	YES
SSH 3.8	YES

UseLogin

This directive is used for controlling interactive login sessions. This is because login does not know how to handle interactive sessions. Enabling this option automatically disables X11Forwarding because it does not know how to handle `xauth` cookies. This option also disables `PermitUserEnvironment` when it is enabled.

Table 42 UseLogin Default Values

SSH Version	Default Values
SSH 3.6	NO
SSH 3.7	NO
SSH 3.8	NO

UsePAM

This is used to provide strict logon security and flexibility in configuring the privileged applications in the system. Thus, PAM uses the interactive keyboard mode instead of using host keys and publickeys, which it does by default. When `PasswordAuthentication` and `UsePAM` are set to yes, the password will be checked for three times in the case of failed password and a new prompt is displayed. This indicates that `ssh` fails back to Password authentication. If this directive is enabled, probably `PasswordAuthentication` is disabled.

Table 43 UsePAM Default Values

SSH Version	Default Values
SSH 3.6	NA
SSH 3.7	Yes*
SSH 3.8	Yes*

UsePrivilegeSeparation

This directive is used to specify whether `sshd` separates privileges by creating an unprivileged child process to deal with incoming network traffic. After successful authentication, another process will be created that has the privilege of the authenticated user. The goal of privilege separation is to prevent privilege escalation by containing any corruption within the unprivileged processes.

Table 44 UsePrivilegeSeparation Default Values

SSH Version	Default Values
SSH 3.6	YES
SSH 3.7	YES
SSH 3.8	YES

X11DisplayOffset

This directive is used to specify the lowest display numbers that `sshd` can use. This will be the first number that `sshd` will use for X11 display and remaining numbers will be greater than this one. This is used to prevent `sshd` crashing the real X11 servers.

Table 45 X11DisplayOffset Default Values

SSH Version	Default Values
SSH 3.6	10
SSH 3.7	10
SSH 3.8	10

X11Forwarding

This directive is used to route X protocol connection through secure Connection and Authentication. When this directive is enabled, there may be additional exposure to the server and to client displays if the sshd proxy display is configured to listen on the wildcard address (see X11UseLocalhost for more details). Additionally, the authentication spoofing and

authentication data verification and substitution occur on the client side. The security risk of using this directive is that the client's X11 display server may be exposed to attack when the ssh client requests forwarding.

Table 46 X11Forwarding Default Values

SSH Version	Default Values
SSH 3.6	YES*
SSH 3.7	YES*
SSH 3.8	YES*

X11UseLocalhost

This directive is used to bound the X11 forwarding server. This directive will let `sshd` to bound the forwarding server either to loopbackaddress or wildcardaddress. In case of loopbackaddress the hostname part of the `DISPLAY` environment variable is localhost.This prevents remote hosts from connecting to the proxy display.

However, some older X11 clients may not function with this configuration. `X11UseLocal` host may be set to no to specify that the forwarding server should be bound to the wildcard address.

Table 47 X11UseLocalhost Default Values

SSH Version	Default Values
SSH 3.6	NO*
SSH 3.7	NO*
SSH 3.8	NO*

HP-UX Secure Shell and chroot-ed environments

HP-UX Secure Shell version A.03.81 supports chroot functionality for both SSH and SFTP. The `chroot` functionality is mainly used as an added security measure. It forces the root directory to become something other than its default.

The `chroot` functionality is a directory jail. It allows an application to start in a specified directory and allows all its users to access that directory and the directories below it. It prevents the users from doing a `cd` above the specified directory. It is intended to allow restricted file and directory access to users of that application while they are in the application. This is not an end-user feature. The system administrator is required to do the necessary setup to enable `chroot` for an application, and all users of that application will automatically be subject to the restrictions imposed by `chroot`. The administrator will need to create new directories, and copy the relevant set of files into the newly created directories, in order for `chroot` to take effect.

New user and chroot configuration

The `chroot` directory location has to be decided for the specific user. This is the directory that the user is restricted within. In the following example, `newroot` is the `chroot` directory, and `appuser` is the sample userid. Follow these steps to set the chroot environment for a new user:

Step 1. Create the new user and group on the system with the chroot directory location as the home directory for the user. Use the following command to create the new user and group:

```
. /usr/sbin/useradd  -c 'chroot user' -d /newroot/./home/appuser
-s /bin/sh  appuser
```

Step 2. Set the password for the new user `appuser`. Once the user is successfully added with the command mentioned in Step 1, the following entry in the `/etc/passwd` file entry for this new user is displayed:

```
appuser:*:120:20:chroot user:/newroot/./home/appuser:/bin/sh
```

NOTE The home directory name will display `a /./` in the name. However, the real home directory will be "/newroot/home/appuser".

Step 3. Create the chroot directory for this user, as shown (In the example, the newroot directory is under `/`). `/sbin/mkdir /newroot/`

Configuring chroot environment for existing users

In the following example, `newroot` is the chroot directory, and `appuser` is a sample userid.

Step 1. Modify the home directory configuration of the user in the `/etc/passwd` file with the `usermod` command as shown below. Alternatively, you can edit the `/etc/passwd` file.

```
 /usr/sbin/usermod  -d /newroot/./home/appuser  -s /bin/sh
appuser
```

Step 2. Create the chroot directory, if it does not already exist, as shown below:

```
/sbin/mkdir /newroot/
```

Step 3. To set up the chroot environment for `sftp`, see “Configuring chroot for sftp” on page 48. For `ssh`, see “Configuring chroot for ssh” on page 51.

Configuring chroot for sftp

Once the new chroot directory has been created follow these steps for `sftp`:

Step 1. Create subdirectories under the new chroot directory. `mkdir -m 755 -p bin /newroot/usr/bin /newroot/usr/lib /newroot/opt/ssh/libexec /newroot/home/appuser`

Step 2. Copy the following required files for the chroot user (`sftp` requires the above libraries in the `/newroot/usr/lib` directory)

For all Platforms

Table 48 Files Required for a chroot User

File	Description
`cp -p /bin/sh/newroot/bin/sh`	Gets the shell for the user.

Table 48 Files Required for a chroot User (Continued)

File	Description
cp -p /usr/bin/cp /usr/bin/ls /usr/bin/mkdir /usr/bin/mv /usr/bin/rm /usr/bin/rmdir /newroot/usr/bin	Used for basic operations such as ls, cp, and so on.
cp -p /opt/ssh/libexec/sftp-server /newroot/opt/ssh/libexec/sftp-server	Looks for the sftp-server in the new chroot directory.
cp /usr/lib/libnss_files.1 /newroot/usr/lib	This file is required for some commands to work within sftp.
cp -p /etc/nsswitch.conf /newroot/etc	This file is required for some commands to work within sftp

If the system is PA-RISC 11.00 or PA-RISC 11i Version 1, copy the following files:

- cp -p /usr/lib/dld.sl /usr/lib/libdld.2 /usr/lib/libk5crypto.sl /usr/lib/libnsl.1 /usr/lib/libxti.2 /usr/lib/libc.2 \ /usr/lib/libgen.2 /usr/lib/libkrb5.sl /usr/lib/libsec.2 \ /usr/lib/libcom_err.sl /usr/lib/libgssapi_krb5.sl/usr/lib/libm.2 \ /usr/lib/libxnet.2 /usr/lib/libcurses.1 /newroot/usr/lib
- cp -p /sbin/sh /newroot/sbin/sh
- cp -p /etc/pam.conf /newroot/etc/pam.conf
- cp -p /usr/lib/libpam.1 /usr/lib/libpsm.1 /usr/lib/security/* /newroot/usr/lib

If the system is 11i Version 2, copy the following files:

- cp -p /usr/lib/hpux32/dld.so /usr/lib/hpux32/libc.so.1 \
- /usr/lib/hpux32/libcom_err.so /usr/lib/hpux32/libdl.so.1 \

- /usr/lib/hpux32/libgen.so.1 /usr/lib/hpux32/libgssapi_krb5.so \
- /usr/lib/hpux32/libk5crypto.so /usr/lib/hpux32/libkrb5.so \
- /usr/lib/hpux32/libm.so.1 /usr/lib/hpux32/libnsl.so.1 \
- /usr/lib/hpux32/libogltls.so /usr/lib/hpux32/libpam.so.1 \
- /usr/lib/hpux32/libsec.so.1 /usr/lib/hpux32/libxcurses.so.1 \
- /usr/lib/hpux32/libxnet.so.1 /usr/lib/hpux32/libxti.so.1 \
- /usr/lib/hpux32/uld.so /usr/lib/libcurses.1 /newroot/usr/lib/hpux32
- cp -p /sbin/sh /newroot/sbin/sh
- cp -p /etc/pam.conf /newroot/etc/pam.conf
- cp -p /usr/lib/security/hpux32/* /newroot/usr/lib/security/hpux32

Step 3. Start the sshd server using the following command:

```
/opt/ssh/sbin/sshd
```

Step 4. Try `sftp` for the chrooted user using the following command:

```
/opt/ssh/bin/sftp  appuser@<hostname
```

Step 5. If the `sftp` connection is successfully done, the chroot functionality can be tested using the following command at the `sftp` prompt:

```
cd /

ls
```

The ls command should list the following subdirectories:

```
bin   home  opt  usr
```

NOTE If `sftp` fails with following message:

```
Request for subsystem  'sftp' failed on channel 0
Couldn't read packet: Connection reset by peer # ldd
/opt/ssh/libexec/sftp-server
```

Ensure that the libraries listed by the ldd command are available under the following directory: `/newroot/usr/lib/`

Configuring chroot for ssh

For chroot to work properly with `ssh`, a proper chroot environment must be created and some binaries and libraries must be copied to the chroot environment. Once the new chroot directory has been created as described in "New user and chroot configuration" on page 47, and "Configuring chroot environment for existing users" on page 48, complete the following steps:

Step 1. Create the following subdirectories under the new chroot directory.

- `mkdir -m 755 -p /newroot/bin /newroot/usr/bin /newroot/usr/lib /newroot/home/appuser \ /newroot/etc /newroot/var/adm /newroot/dev/pts /newroot/tmp chmod 666 /newroot/tmp`

Step 2. Copy the required files for the chroot user into these newly created subdirectories

- `cp -pR /dev/pts/* /newroot/dev/pts/`
- `cp -pR /dev/tty /newroot/dev/`

  `ssh` requires the pseudo device to be available in the newroot environment.

- `cp -p /bin/sh /newroot/bin/sh`

  Gets the shell for the user.

- `cp -p /var/adm/wtmp /newroot/var/adm`

  `ssh` requires the wtmp file to be available in the `/newroot/var/adm` directory,

- `cp -p /etc/passwd /newroot/etc/passwd`

  `ssh` requires the `/etc/passwd` file to be available in the `/newroot/etc` directory,

- `cp -p /usr/bin/cp /usr/bin/ls /usr/bin/mkdir /usr/bin/mv /usr/bin/rm \ /usr/bin/rmdir /newroot/usr/bin`

  Used for basic operations such as `ls`, `cp`, and so on.

If the system is PA-RISC running HP-UX 11.00 or 11i Version 1, copy the following libraries:

- cp -p /usr/lib/dld.sl /usr/lib/libdld.2 /usr/lib/libk5crypto.sl /usr/lib/libnsl.1 /usr/lib/libxti.2 /usr/lib/libc.2 \ /usr/lib/libgen.2 /usr/lib/libkrb5.sl /usr/lib/libsec.2 \ /usr/lib/libcom_err.sl /usr/lib/libgssapi_krb5.sl/usr/lib/libm.2 \ /usr/lib/libxnet.2 /usr/lib/libcurses.1 /newroot/usr/lib
- cp -p /sbin/sh /newroot/sbin/sh
- cp -p /etc/pam.conf /newroot/etc/pam.conf
- cp -p /usr/lib/libpam.1 /usr/lib/libpsm.1 /usr/lib/security/* /newroot/usr/lib

If the system is Itanium running HP-UX 11i Version 2, copy the following additional libraries (ssh requires these in the `/newroot/usr/lib` directory):

- cp -p /usr/lib/hpux32/dld.so /usr/lib/hpux32/libc.so.1 /usr/lib/hpux32/libcom_err.so \
- /usr/lib/hpux32/libdl.so.1 /usr/lib/hpux32/libgen.so.1 \
- /usr/lib/hpux32/libgssapi_krb5.so /usr/lib/hpux32/libk5crypto.so \
- /usr/lib/hpux32/libnsl.so.1 /usr/lib/hpux32/libogltls.so \
- /usr/lib/hpux32/libpam.so.1 /usr/lib/hpux32/libsec.so.1
- /usr/lib/hpux32/libxcurses.so.1 /usr/lib/hpux32/libxnet.so.1 \
- /usr/lib/hpux32/libxti.so.1 /usr/lib/hpux32/uld.so /usr/lib/libcurses.1 \
- /newroot/usr/lib/hpux32
- cp -p /sbin/sh /newroot/sbin/sh
- cp -p /etc/pam.conf /newroot/etc/pam.conf
- cp -p /usr/lib/security/hpux32/* /newroot/usr/lib/security/hpux32

Step 3. Start the sshd server using the following command:

```
/opt/ssh/sbin/sshd
```

Step 4. Try `sftp` for the chrooted user.

```
/opt/ssh/bin/ssh appuser@<hostname
```

Step 5. If the sftp connection is successfully done, the chroot functionality can be tested using the following command in the ssh shell:

```
cd /
ls
```

The ls command should list the following subdirectories:

```
bin   home usr  etc  tmp  var  dev
```

NOTE The administrator must set the password for the user "appuser".
