Secure Shell Version 2 Support

The Secure Shell Version 2 Support feature allows you to configure Secure Shell (SSH) Version 2. (SSH Version 1 support was implemented in an earlier Cisco software release.) SSH runs on top of a reliable transport layer and provides strong authentication and encryption capabilities. The only reliable transport that is defined for SSH is TCP. SSH provides a means to securely access and securely execute commands on another computer over a network. The Secure Copy Protocol (SCP) feature that is provided with SSH allows for the secure transfer of files.

- Prerequisites for Secure Shell Version 2 Support, on page 1
- Restrictions for Secure Shell Version 2 Support, on page 2
- Information About Secure Shell Version 2 Support, on page 2
- How to Configure Secure Shell, on page 5
- Configuration Examples for Secure Shell Version 2 Support, on page 17
- Additional References for Secure Shell Version 2 Support, on page 21
- Feature History for Secure Shell Version 2 Support, on page 22

Prerequisites for Secure Shell Version 2 Support

- Before configuring SSH, ensure that the required image is loaded on your device. The SSH server requires you to have a k9 (Triple Data Encryption Standard [3DES]) software image depending on your release.
- You have to use a SSH remote device that supports SSH Version 2 and connect to a Cisco device.
- SCP relies on authentication, authorization, and accounting (AAA) to function correctly. Therefore, AAA must be configured on the device to enable the secure copy protocol on the SSH Server.

Note

The SSH Version 2 server and the SSH Version 2 client are supported on your Cisco software, depending on your release. (The SSH client runs both the SSH Version 1 protocol and the SSH Version 2 protocol. The SSH client is supported in k9 images depending on your release.)
Restrictions for Secure Shell Version 2 Support

- Secure Shell (SSH) servers and SSH clients are supported in Triple Data Encryption Standard (3DES) software images.
- Execution Shell, remote command execution, and Secure Copy Protocol (SCP) are the only applications supported.
- Rivest, Shamir, and Adleman (RSA) key generation is an SSH server-side requirement. Devices that act as SSH clients need not generate RSA keys.
- The RSA key pair size must be greater than or equal to 768 bits.
- The following features are not supported:
  - Port forwarding
  - Compression

Information About Secure Shell Version 2 Support

Secure Shell Version 2


The configuration for the SSH Version 2 server is similar to the configuration for SSH Version 1. The `ip ssh version` command defines the SSH version to be configured. If you do not configure this command, SSH by default runs in compatibility mode; that is, both SSH Version 1 and SSH Version 2 connections are honored.

SSH Version 1 is a protocol that has never been defined in a standard. If you do not want your device to fall back to the undefined protocol (Version 1), you should use the `ip ssh version` command and specify Version 2.

The `ip ssh rsa keypair-name` command enables an SSH connection using the Rivest, Shamir, and Adleman (RSA) keys that you have configured. Previously, SSH was linked to the first RSA keys that were generated (that is, SSH was enabled when the first RSA key pair was generated). This behavior still exists, but by using the `ip ssh rsa keypair-name` command, you can overcome this behavior. If you configure the `ip ssh rsa keypair-name` command with a key pair name, SSH is enabled if the key pair exists or SSH will be enabled if the key pair is generated later. If you use this command to enable SSH, you are not forced to configure a hostname and a domain name, which was required in SSH Version 1 of the Cisco software.

The login banner is supported in SSH Version 2, but it is not supported in Secure Shell Version 1.
Secure Shell Version 2 Enhancements

The SSH Version 2 Enhancements feature includes a number of additional capabilities such as supporting Virtual Routing and Forwarding (VRF)-Aware SSH, SSH debug enhancements, and Diffie-Hellman (DH) group exchange support.

Note

The VRF-Aware SSH feature is supported depending on your release.

The Cisco SSH implementation has traditionally used 768-bit modulus, but with an increasing need for higher key sizes to accommodate DH Group 14 (2048 bits) and Group 16 (4096 bits) cryptographic applications, a message exchange between the client and the server to establish the favored DH group becomes necessary. The `ip ssh dh min size` command configures the modulus size on the SSH server. In addition to this, the `ssh` command was extended to add VRF awareness to the SSH client-side functionality through which the VRF instance name in the client is provided with the IP address to look up the correct routing table and establish a connection.

Debugging was enhanced by modifying SSH debug commands. The `debug ip ssh` command was extended to simplify the debugging process. Before the simplification of the debugging process, this command printed all debug messages related to SSH regardless of what was specifically required. The behavior still exists, but if you configure the `debug ip ssh` command with a keyword, messages are limited to information specified by the keyword.

Secure Shell Version 2 Enhancements for RSA Keys

Cisco SSH Version 2 supports keyboard-interactive and password-based authentication methods. The SSH Version 2 Enhancements for RSA Keys feature also supports RSA-based public key authentication for the client and the server.

User authentication—RSA-based user authentication uses a private/public key pair associated with each user for authentication. The user must generate a private/public key pair on the client and configure a public key on the Cisco SSH server to complete the authentication.

An SSH user trying to establish credentials provides an encrypted signature using the private key. The signature and the user’s public key are sent to the SSH server for authentication. The SSH server computes a hash over the public key provided by the user. The hash is used to determine if the server has a matching entry. If a match is found, an RSA-based message verification is performed using the public key. Hence, the user is authenticated or denied access based on the encrypted signature.

Server authentication—While establishing an SSH session, the Cisco SSH client authenticates the SSH server by using the server host keys available during the key exchange phase. SSH server keys are used to identify the SSH server. These keys are created at the time of enabling SSH and must be configured on the client.

For server authentication, the Cisco SSH client must assign a host key for each server. When the client tries to establish an SSH session with a server, the client receives the signature of the server as part of the key exchange message. If the strict host key checking flag is enabled on the client, the client checks if it has the host key entry corresponding to the server. If a match is found, the client tries to validate the signature by using the server host key. If the server is successfully authenticated, the session establishment continues; otherwise, it is terminated and displays a “Server Authentication Failed” message.
Storing public keys on a server uses memory; therefore, the number of public keys configurable on an SSH server is restricted to ten users, with a maximum of two public keys per user.

Note
RSA-based user authentication is supported by the Cisco server, but Cisco clients cannot propose public key as an authentication method. If the Cisco server receives a request from an open SSH client for RSA-based authentication, the server accepts the authentication request.

Note
For server authentication, configure the RSA public key of the server manually and configure the `ip ssh stricthostkeycheck` command on the Cisco SSH client.

**SSH And Switch Access**

Secure Shell (SSH) is a protocol that provides a secure, remote connection to a device. SSH provides more security for remote connections than Telnet does by providing strong encryption when a device is authenticated. This software release supports SSH Version 2 (SSHv2).

SSH functions the same in IPv6 as in IPv4. For IPv6, SSH supports IPv6 addresses and enables secure, encrypted connections with remote IPv6 nodes over an IPv6 transport.

**SNMP Trap Generation**

Depending on your release, Simple Network Management Protocol (SNMP) traps are generated automatically when an SSH session terminates if the traps have been enabled and SNMP debugging has been enabled.

Note
When you configure the `snmp-server host` command, the IP address must be the address of the PC that has the SSH (telnet) client and that has IP connectivity to the SSH server.

You must also enable SNMP debugging using the `debug snmp packet` command to display the traps. The trap information includes information such as the number of bytes sent and the protocol that was used for the SSH session.

**SSH Keyboard Interactive Authentication**

The SSH Keyboard Interactive Authentication feature, also known as Generic Message Authentication for SSH, is a method that can be used to implement different types of authentication mechanisms. Basically, any currently supported authentication method that requires only user input can be performed with this feature. The feature is automatically enabled.

The following methods are supported:

- Password
• SecurID and hardware tokens printing a number or a string in response to a challenge sent by the server
• Pluggable Authentication Module (PAM)
• S/KEY (and other One-Time-Pads)

How to Configure Secure Shell

Configuring a Device for SSH Version 2 Using a Hostname and Domain Name

Procedure

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> hostname name</td>
<td>Configures a hostname for your device.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# hostname catalyst9k</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> ip domain name name</td>
<td>Configures a domain name for your device.</td>
</tr>
<tr>
<td><strong>Example:</strong> catalyst9k(config)# ip domain name example.com</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> crypto key generate rsa</td>
<td>Enables the SSH server for local and remote authentication.</td>
</tr>
<tr>
<td><strong>Example:</strong> catalyst9k(config)# crypto key generate rsa</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> ip ssh [time-out seconds</td>
<td>(Optional) Configures SSH control variables on your device.</td>
</tr>
<tr>
<td>authentication-retries integer]</td>
<td><strong>Example:</strong> catalyst9k(config)# ip ssh time-out 120</td>
</tr>
</tbody>
</table>
### Configuring a Device for SSH Version 2 Using RSA Key Pairs

**Procedure**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> ip ssh rsa keypair-name <strong>keypair-name</strong></td>
<td>Specifies the RSA key pair to be used for SSH.</td>
</tr>
<tr>
<td>Example: Device(config)# ip ssh rsa keypair-name sshkeys</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> crypto key generate rsa usage-keys label <strong>key-label</strong> modulus <strong>modulus-size</strong></td>
<td>Enables the SSH server for local and remote authentication on the device.</td>
</tr>
<tr>
<td>Example: Device(config)# crypto key generate rsa usage-keys label sshkeys modulus 768</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** A Cisco device can have many RSA key pairs.

**Note:** To delete the RSA key pair, use the `crypto key zeroize rsa` command. When you delete the RSA key pair, you automatically disable the SSH server.

**Secure Shell Version 2 Support**

---

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 7</strong> ip ssh version [1</td>
<td>2]</td>
</tr>
<tr>
<td>Example: catalyst9k(config)# ip ssh version 1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> exit</td>
<td>Exits global configuration mode and enters privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: catalyst9k(config)# exit</td>
<td></td>
</tr>
</tbody>
</table>

- Use `no hostname` command to return to the default host.

---

### Secure Shell Version 2 Support

---
### Configuring the Cisco SSH Server to Perform RSA-Based User Authentication

#### Procedure

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1**
  enable
  Example:
  Device> enable | Enables privileged EXEC mode.
  * Enter your password if prompted. |
| **Step 2**
  configure terminal
  Example:
  Device# configure terminal | Enters global configuration mode. |
| **Step 3**
  hostname name
  Example:
  Device(config)# hostname host1 | Specifies the hostname. |
| **Step 4**
  ip domain name name
  Example:
  host1(config)# ip domain name name1 | Defines a default domain name that the Cisco software uses to complete unqualified hostnames. |
| **Step 5**
  crypto key generate rsa
  Example:
  host1(config)# crypto key generate rsa | Generates RSA key pairs. |
## Configuring the Cisco SSH Server to Perform RSA-Based User Authentication

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td>ip ssh pubkey-chain</td>
<td>Configures SSH-RSA keys for user and server authentication on the SSH server and enters public-key configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• The user authentication is successful if the RSA public key stored on the server is verified with the public or the private key pair stored on the client.</td>
</tr>
<tr>
<td>host1(config)# ip ssh pubkey-chain</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td>username username</td>
<td>Configures the SSH username and enters public-key user configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>host1(conf-ssh-pubkey)# username user1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td>key-string</td>
<td>Specifies the RSA public key of the remote peer and enters public-key data configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Note You can obtain the public key value from an open SSH client; that is, from the .ssh/id_rsa.pub file.</td>
</tr>
<tr>
<td>host1(conf-ssh-pubkey-user)# key-string</td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td>key-hash key-type key-name</td>
<td>(Optional) Specifies the SSH key type and version.</td>
</tr>
<tr>
<td>Example:</td>
<td>• The key type must be ssh-rsa for the configuration of private public key pairs.</td>
</tr>
<tr>
<td>host1(conf-ssh-pubkey-data)# key-hash ssh-rsa key1</td>
<td>• This step is optional only if the key-string command is configured.</td>
</tr>
<tr>
<td></td>
<td>• You must configure either the key-string command or the key-hash command.</td>
</tr>
<tr>
<td></td>
<td>Note You can use a hashing software to compute the hash of the public key string, or you can also copy the hash value from another Cisco device. Entering the public key data using the key-string command is the preferred way to enter the public key data for the first time.</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td>end</td>
<td>Exits public-key data configuration mode and returns to privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Use no hostname command to return to the default host.</td>
</tr>
<tr>
<td>host1(conf-ssh-pubkey-data)# end</td>
<td></td>
</tr>
</tbody>
</table>
# Configuring the Cisco IOS SSH Client to Perform RSA-Based Server Authentication

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><code>enable</code></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device&gt; <code>enable</code></td>
<td>Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><code>configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device# <code>configure terminal</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><code>hostname name</code></td>
<td>Specifies the hostname.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config)# <code>hostname host1</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><code>ip domain name name</code></td>
<td>Defines a default domain name that the Cisco software uses to complete unqualified hostnames.</td>
</tr>
<tr>
<td>Example:</td>
<td>host1(config)# <code>ip domain name name1</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><code>crypto key generate rsa</code></td>
<td>Generates RSA key pairs.</td>
</tr>
<tr>
<td>Example:</td>
<td>host1(config)# <code>crypto key generate rsa</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><code>ip ssh pubkey-chain</code></td>
<td>Configures SSH-RSA keys for user and server authentication on the SSH server and enters public-key configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>host1(config)# <code>ip ssh pubkey-chain</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><code>server server-name</code></td>
<td>Enables the SSH server for public-key authentication on the device and enters public-key server configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>host1(conf-ssh-pubkey)# <code>server server1</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td><code>key-string</code></td>
<td>Specifies the RSA public-key of the remote peer and enters public key data configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>host1(conf-ssh-pubkey-server)# <code>key-string</code></td>
<td><strong>Note</strong> You can obtain the public key value from an open SSH client; that is, from the .ssh/id_rsa.pub file.</td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>exit</td>
<td>Exits public-key data configuration mode and enters public-key server configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>host1(conf-ssh-pubkey-data)# exit</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>key-hash key-type key-name</td>
<td>(Optional) Specifies the SSH key type and version.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>host1(conf-ssh-pubkey-server)# key-hash ssh-rsa key1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The key type must be ssh-rsa for the configuration of private/public key pairs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• This step is optional only if the key-string command is configured.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• You must configure either the key-string command or the key-hash command.</td>
</tr>
<tr>
<td></td>
<td>Note</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>You can use a hashing software to compute the hash of the public key string, or you can copy the hash value from another Cisco device. Entering the public key data using the key-string command is the preferred way to enter the public key data for the first time.</td>
</tr>
<tr>
<td>11</td>
<td>end</td>
<td>Exits public-key server configuration mode and returns to privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>host1(conf-ssh-pubkey-server)# end</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>host1# configure terminal</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>ip ssh stricthostkeycheck</td>
<td>Ensures that server authentication takes place.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>host1(config)# ip ssh stricthostkeycheck</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The connection is terminated in case of a failure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Use no hostname command to return to the default host.</td>
</tr>
<tr>
<td>14</td>
<td>end</td>
<td>Exits global configuration mode and returns to privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>host1(config)# end</td>
<td></td>
</tr>
</tbody>
</table>
Starting an Encrypted Session with a Remote Device

The device with which you want to connect must support a Secure Shell (SSH) server that has an encryption algorithm that is supported in Cisco software. Also, you need not enable your device. SSH can be run in disabled mode.

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td></td>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>ssh [-v {1</td>
<td>2}]</td>
</tr>
<tr>
<td></td>
<td>-c {aes128-ctr</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>aes192-ctr</td>
</tr>
<tr>
<td></td>
<td></td>
<td>aes256-ctr</td>
</tr>
<tr>
<td></td>
<td></td>
<td>aes128-cbc</td>
</tr>
<tr>
<td></td>
<td></td>
<td>aes192-cbc</td>
</tr>
<tr>
<td></td>
<td></td>
<td>aes256-cbc</td>
</tr>
<tr>
<td></td>
<td>-l user-id</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-l user-id:vrf-name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>number ip-address</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ip-address</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-l ip-addr</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-l hostname</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[command</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-vrf]</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>Device# ssh -v 2 -c aes256-ctr -m hmac-sha1-96 -l user2 10.76.82.24</td>
<td></td>
</tr>
</tbody>
</table>

Verifying the Status of the Secure Shell Connection

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td></td>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>show ssh</td>
<td>Displays the status of SSH server connections.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device# show ssh</td>
<td></td>
</tr>
</tbody>
</table>
### Verifying the Secure Shell Version 2 Status

#### Procedure

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
</tbody>
</table>

The following sample output from the `show ssh` command displays status of various SSH Version 1 and Version 2 connections for Version 1 and Version 2 connections:

```
Device# show ssh
Connection  Version  Encryption  State     Username
  0        1.5       3DES       Session started  lab
Connection  Version  Mode  Encryption  Hmac  State     Username
  1        2.0       IN    aes128-cbc  hmac-md5 Session started  lab
  1        2.0       OUT   aes128-cbc  hmac-md5 Session started  lab
```

The following sample output from the `show ssh` command displays status of various SSH Version 1 and Version 2 connections for a Version 2 connection with no Version 1 connection:

```
Device# show ssh
Connection  Version  Mode  Encryption  Hmac  State     Username
  1        2.0       IN    aes128-cbc  hmac-md5 Session started  lab
  1        2.0       OUT   aes128-cbc  hmac-md5 Session started  lab
%No SSHv1 server connections running.
```

The following sample output from the `show ssh` command displays status of various SSH Version 1 and Version 2 connections for a Version 1 connection with no Version 2 connection:

```
Device# show ssh
Connection  Version  Encryption  State     Username
  0        1.5       3DES       Session started  lab
%No SSHv2 server connections running.
```
### Command or Action

**Example:**

Device> enable

**Purpose:**

- Enter your password if prompted.

#### Step 2

**show ip ssh**

**Example:**

Device# show ip ssh

**Purpose:**

Displays the version and configuration data for SSH.

#### Step 3

**exit**

**Example:**

Device# exit

**Purpose:**

Exits privileged EXEC mode and returns to user EXEC mode.

---

### Examples

The following sample output from the `show ip ssh` command displays the version of SSH that is enabled, the authentication timeout values, and the number of authentication retries for Version 1 and Version 2 connections:

```
Device# show ip ssh
SSH Enabled - version 1.99
Authentication timeout: 120 secs; Authentication retries: 3
```

The following sample output from the `show ip ssh` command displays the version of SSH that is enabled, the authentication timeout values, and the number of authentication retries for a Version 2 connection with no Version 1 connection:

```
Device# show ip ssh
SSH Enabled - version 2.0
Authentication timeout: 120 secs; Authentication retries: 3
```

The following sample output from the `show ip ssh` command displays the version of SSH that is enabled, the authentication timeout values, and the number of authentication retries for a Version 1 connection with no Version 2 connection:

```
Device# show ip ssh
3d06h: %SYS-5-CONFIG_I: Configured from console by console
SSH Enabled - version 1.5
Authentication timeout: 120 secs; Authentication retries: 3
```
## Monitoring and Maintaining Secure Shell Version 2

### Procedure

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| 1    | `enable`          | Enables privileged EXEC mode.  
     |                  | • Enter your password if prompted. |
|      | **Example:**      |         |
|      | Device> `enable`  |         |
| 2    | `debug ip ssh`   | Enables debugging of SSH. |
|      | **Example:**      |         |
|      | Device# `debug ip ssh` |     |
| 3    | `debug snmp packet` | Enables debugging of every SNMP packet sent or received by the device. |
|      | **Example:**      |         |
|      | Device# `debug snmp packet` | |

### Example

The following sample output from the `debug ip ssh` command shows the connection is an SSH Version 2 connection:

```
Device# `debug ip ssh`
00:33:55: SSH1: starting SSH control process
00:33:55: SSH1: sent protocol version id SSH-1.99-Cisco-1.25
00:33:55: SSH1: protocol version id is - SSH-2.0-OpenSSH_2.5.2p2
00:33:55: SSH2 1: send: len 280 (includes padlen 4)
00:33:55: SSH2 1: SSH2_MSG_KEXINIT sent
00:33:55: SSH2 1: ssh_receive: 536 bytes received
00:33:55: SSH2 1: input: packet len 632
00:33:55: SSH2 1: partial packet 8, need 624, maclen 0
00:33:55: SSH2 1: ssh_receive: 96 bytes received
00:33:55: SSH2 1: partial packet 8, need 624, maclen 0
00:33:55: SSH2 1: input: padlen 11
00:33:55: SSH2 1: received packet type 20
00:33:55: SSH2 1: SSH2_MSG_KEXINIT received
00:33:55: SSH2: kex: client->server aes128-cbc hmac-md5 none
00:33:55: SSH2: kex: server->client aes128-cbc hmac-md5 none
00:33:55: SSH2 1: expecting SSH2_MSG_KEXDH_INIT
00:33:55: SSH2 1: ssh_receive: 144 bytes received
00:33:55: SSH2 1: input: packet len 144
00:33:55: SSH2 1: partial packet 8, need 136, maclen 0
00:33:55: SSH2 1: input: padlen 5
00:33:55: SSH2 1: received packet type 30
00:33:55: SSH2 1: SSH2_MSG_KEXDH_INIT received
00:33:55: SSH2 1: signature length 111
00:33:55: SSH2 1: send: len 384 (includes padlen 7)
00:33:55: SSH2 1: kex_derive_keys complete
00:33:55: SSH2 1: send: len 16 (includes padlen 10)
```
Secure Shell Version 2 Support
Secure Shell Version 2 Support

00:34:04: SSH2 1: input: packet len 32
00:34:04: SSH2 1: partial packet 16, need 16, maclen 16
00:34:04: SSH2 1: MAC #10 ok
00:34:04: SSH2 1: input: padlen 12
00:34:04: SSH2 1: received packet type 98
00:34:04: SSH2 1: shell request
00:34:04: SSH2 1: shell message received
00:34:04: SSH2 1: starting shell for vty
00:34:04: SSH2 1: send: len 48 (includes padlen 18)
00:34:04: SSH2 1: done calc MAC out #7
00:34:07: SSH2 1: ssh_receive: 48 bytes received
00:34:07: SSH2 1: input: packet len 32
00:34:07: SSH2 1: partial packet 16, need 16, maclen 16
00:34:07: SSH2 1: MAC #11 ok
00:34:07: SSH2 1: input: padlen 17
00:34:07: SSH2 1: received packet type 94
00:34:07: SSH2 1: send: len 32 (includes padlen 17)
00:34:07: SSH2 1: done calc MAC out #8
00:34:07: SSH2 1: ssh_receive: 48 bytes received
00:34:07: SSH2 1: input: packet len 32
00:34:07: SSH2 1: partial packet 16, need 16, maclen 16
00:34:07: SSH2 1: MAC #12 ok
00:34:07: SSH2 1: input: padlen 17
00:34:07: SSH2 1: received packet type 94
00:34:07: SSH2 1: send: len 32 (includes padlen 17)
00:34:07: SSH2 1: done calc MAC out #9
00:34:07: SSH2 1: ssh_receive: 48 bytes received
00:34:07: SSH2 1: input: packet len 32
00:34:07: SSH2 1: partial packet 16, need 16, maclen 16
00:34:07: SSH2 1: MAC #13 ok
00:34:07: SSH2 1: input: padlen 17
00:34:07: SSH2 1: received packet type 94
00:34:08: SSH2 1: send: len 32 (includes padlen 17)
00:34:08: SSH2 1: done calc MAC out #10
00:34:08: SSH2 1: ssh_receive: 48 bytes received
00:34:08: SSH2 1: input: packet len 32
00:34:08: SSH2 1: partial packet 16, need 16, maclen 16
00:34:08: SSH2 1: MAC #14 ok
00:34:08: SSH2 1: input: padlen 17
00:34:08: SSH2 1: received packet type 94
00:34:08: SSH2 1: send: len 32 (includes padlen 17)
00:34:08: SSH2 1: done calc MAC out #11
00:34:08: SSH2 1: ssh_receive: 48 bytes received
00:34:08: SSH2 1: input: packet len 32
00:34:08: SSH2 1: partial packet 16, need 16, maclen 16
00:34:08: SSH2 1: MAC #15 ok
00:34:08: SSH2 1: input: padlen 17
00:34:08: SSH2 1: received packet type 94
00:34:08: SSH2 1: send: len 32 (includes padlen 16)
00:34:08: SSH2 1: done calc MAC out #12
00:34:08: SSH2 1: send: len 48 (includes padlen 18)
00:34:08: SSH2 1: done calc MAC out #13
00:34:08: SSH2 1: send: len 16 (includes padlen 6)
00:34:08: SSH2 1: done calc MAC out #14
00:34:08: SSH2 1: send: len 16 (includes padlen 6)
00:34:08: SSH2 1: done calc MAC out #15
00:34:08: SSH1: Session terminated normally
Configuration Examples for Secure Shell Version 2 Support

Example: Configuring Secure Shell Version 2

```
Device> enable
Device# configure terminal
Device(config)# ip ssh version 2
Device(config)# end
```

Example: Configuring Secure Shell Versions 1 and 2

```
Device> enable
Device# configure terminal
Device(config)# no ip ssh version
Device(config)# end
```

Example: Starting an Encrypted Session with a Remote Device

```
Device> enable
Device# ssh -v 2 -c aes256-cbc -m hmac-shal-160 -l shaship 10.76.82.24
```

Example: Setting an SNMP Trap

```
The following example shows how to set an SNMP trap is set. The trap notification is generated automatically when the SSH session terminates. In the example, 10.1.1.1 is the IP address of the SSH client.

Device> enable
Device# configure terminal
Device(config)# snmp-server trap link switchover
Device(config)# snmp-server host 10.1.1.1 public tty
Device(config)# end
```

Examples: SSH Keyboard Interactive Authentication

Example: Enabling Client-Side Deubs

```
The following example shows that the client-side deubs are turned on, and the maximum number of prompts is six (three for the SSH keyboard interactive authentication method and three for the password authentication method).
```

Password:
Password:
Password:
Password:
Password:
Password: cisco123
Last login: Tue Dec 6 13:15:21 2005 from 10.76.248.213
Example: Enabling ChPass with a Blank Password Change

In the following example, the ChPass feature is enabled, and a blank password change is accomplished using the SSH Keyboard Interactive Authentication method. A TACACS+ access control server (ACS) is used as the back-end AAA server.

Device> enable
Device1# ssh -l cisco 10.1.1.3

Password:
Old Password: cisco
New Password: cisco123
Re-enter New password: cisco123

Example: Enabling ChPass with a Blank Password Change

Example: Enabling ChPass with a Blank Password Change

In the following example, the ChPass feature is enabled, and a blank password change is accomplished using the SSH Keyboard Interactive Authentication method. A TACACS+ access control server (ACS) is used as the back-end AAA server.

Device> enable
Device1# ssh -l cisco 10.1.1.3

Password:
Old Password: cisco
New Password: cisco123
Re-enter New password: cisco123

Example: Enabling ChPass and Changing the Password on First Login

In the following example, the ChPass feature is enabled and TACACS+ ACS is used as the back-end server. The password is changed on the first login using the SSH keyboard interactive authentication method.

Device1> enable
Device1# ssh -l cisco 10.1.1.3

Password: cisco
Your password has expired.
Enter a new one now.
New Password: cisco123
Re-enter New password: cisco123
Device2> exit
[Connection to 10.1.1.3 closed by foreign host]

Device1# ssh -l cisco 10.1.1.3
Password: cisco
Your password has expired.
Enter a new one now.
New Password: cisco
Re-enter New password: cisco
The New and Re-entered passwords have to be the same.
Try again.
New Password: cisco
Re-enter New password: cisco

Device2>

Example: Enabling ChPass and Expiring the Password After Three Logins

In the following example, the ChPass feature is enabled and TACACS+ ACS is used as the back-end AAA server. The password expires after three logins using the SSH keyboard interactive authentication method.

Device# ssh -l cisco 10.1.1.3
Password: cisco
Device2> exit
[Connection to 10.1.1.3 closed by foreign host]

Device1# ssh -l cisco 10.1.1.3
Password: cisco
Device2> exit
Device1# ssh -l cisco 10.1.1.3
Password: cisco
Device2> exit
Device1# ssh -l cisco 10.1.1.3
Password: cisco
Your password has expired.
Enter a new one now.
New Password: cisco123
Re-enter New password: cisco123

Device2>

Example: SNMP Debugging

The following is sample output from the debug snmp packet command. The output provides SNMP trap information for an SSH session.
Device1# debug snmp packet
SNMP packet debugging is on
Device1# ssh -l lab 10.0.0.2
Password:
Device2# exit
[Connection to 10.0.0.2 closed by foreign host]
Device1#
*Jul 18 10:18:42.619: SNMP: Queuing packet to 10.0.0.2
*Jul 18 10:18:42.619: SNMP: V1 Trap, ent cisco, addr 10.0.0.1, gentrap 6, spectrap 1
local.9.3.1.1.2.1 = 6
tcpConnEntry.1.10.0.0.1.22.10.0.0.2.55246 = 4
ltcpConnEntry.5.10.0.0.1.22.10.0.0.2.55246 = 1015
ltcpConnEntry.1.10.0.0.1.22.10.0.0.2.55246 = 1056
ltcpConnEntry.2.10.0.0.1.22.10.0.0.2.55246 = 1392
local.9.2.1.18.2 = lab
*Jul 18 10:18:42.879: SNMP: Packet sent via UDP to 10.0.0.2
Device1#

Examples: SSH Debugging Enhancements

The following is sample output from the debug ip ssh detail command. The output provides debugging information about the SSH protocol and channel requests.

Device1# debug ip ssh detail
00:04:22: SSH0: starting SSH control process
00:04:22: SSH0: sent protocol version id SSH-1.99-Cisco-1.25
00:04:22: SSH0: protocol version id is - SSH-1.99-Cisco-1.25
00:04:22: SSH2 0: SSH2_MSG_KEXINIT sent
00:04:22: SSH2 0: SSH2_MSG_KEXINIT received
00:04:22: SSH2 0: expecting SSH2_MSG_KEXDH_INIT
00:04:22: SSH2 0: SSH2_MSG_KEXDH_INIT received
00:04:22: SSH2 0: kex_derive_keys complete
00:04:22: SSH2 0: SSH2_MSG_NEWKEYS sent
00:04:22: SSH2 0: SSH2_MSG_NEWKEYS received
00:04:22: SSH2 0: authentication successful for lab
00:04:22: SSH2 0: channel open request
00:04:24: SSH2 0: pty-req request
00:04:24: SSH2 0: setting TTY - requested: height 24, width 80; set: height 24, width 80
00:04:24: SSH2 0: shell request
00:04:24: SSH2 0: shell message received
00:04:24: SSH2 0: starting shell for vty
00:04:38: SSH0: Session terminated normally

The following is sample output from the debug ip ssh packet command. The output provides debugging information about the SSH packet.

Device1# debug ip ssh packet
00:05:43: SSH2 0: send:packet of length 280 (length also includes padlen of 4)
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: input: total packet length of 280 bytes
00:05:43: SSH2 0: partial packet length(block size) 8 bytes, needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size) 8 bytes, needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size) 8 bytes, needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 24 bytes received
00:05:43: SSH2 0: partial packet length(block size) 8 bytes, needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: input: padlength 4 bytes
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size) 8 bytes, needed 136 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size) 8 bytes, needed 136 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 16 bytes received
00:05:43: SSH2 0: partial packet length(block size) 8 bytes, needed 136 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 16 bytes received
00:05:43: SSH2 0: input: padlength 6 bytes
00:05:43: SSH2 0: input: total packet length of 144 bytes
00:05:43: SSH2 0: partial packet length(block size) 16 bytes, needed 16 bytes, maclen 20
00:05:43: SSH2 0: MAC compared for #3: ok

## Additional References for Secure Shell Version 2 Support

### Related Documents

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS commands</td>
<td>Cisco IOS Master Command List, All Releases</td>
</tr>
<tr>
<td>SSH Version 1</td>
<td>Configuring Secure Shell chapter of the Security Configuration Guide</td>
</tr>
</tbody>
</table>

### Standards

<table>
<thead>
<tr>
<th>Standards</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>IETF Secure Shell Version 2 Draft Standards</td>
<td>Internet Engineering Task Force website</td>
</tr>
</tbody>
</table>
Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Cisco Support website provides extensive online resources, including</td>
<td><a href="http://www.cisco.com/support">http://www.cisco.com/support</a></td>
</tr>
<tr>
<td>documentation and tools for troubleshooting and resolving technical issues</td>
<td></td>
</tr>
<tr>
<td>with Cisco products and technologies.</td>
<td></td>
</tr>
<tr>
<td>To receive security and technical information about your products, you can</td>
<td></td>
</tr>
<tr>
<td>subscribe to various services, such as the Product Alert Tool (accessed from</td>
<td></td>
</tr>
<tr>
<td>Field Notices), the Cisco Technical Services Newsletter, and Really Simple</td>
<td></td>
</tr>
<tr>
<td>Syndication (RSS) Feeds.</td>
<td></td>
</tr>
<tr>
<td>Access to most tools on the Cisco Support website requires a Cisco.com user</td>
<td></td>
</tr>
<tr>
<td>ID and password.</td>
<td></td>
</tr>
</tbody>
</table>

Feature History for Secure Shell Version 2 Support

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

<table>
<thead>
<tr>
<th>Release</th>
<th>Feature</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS XE Everest 16.5.1a</td>
<td>Secure Shell Version 2 Support</td>
<td>The Secure Shell Version 2 Support feature allows you to configure Secure Shell (SSH) Version 2 (SSH Version 1 support was implemented in an earlier Cisco IOS software release). SSH runs on top of a reliable transport layer and provides strong authentication and encryption capabilities. SSH version 2 also supports AES counter-based encryption mode. Support for this feature was introduced on all the models of the Cisco Catalyst 9500 Series Switches.</td>
</tr>
</tbody>
</table>

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to [http://www.cisco.com/go/cfn](http://www.cisco.com/go/cfn).