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.

- Finding Feature Information, on page 1
- Information About Secure Shell Version 2 Support, on page 1
- How to Configure Secure Shell Version 2 Support, on page 5
- Configuration Examples for Secure Shell Version 2 Support, on page 19
- Additional References for Secure Shell Version 2 Support, on page 24
- Feature Information for Secure Shell Version 2 Support, on page 25

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

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.

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.

Note: 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.

**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. For information about enabling SNMP traps, see the “Configuring SNMP Support” module in the **SNMP Configuration Guide**.

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.

The following example shows that an SNMP trap is set. The trap notification is generated automatically when the SSH session terminates. In the example, a.b.c.d is the IP address of the SSH client.

```
snmp-server
snmp-server host a.b.c.d public tty
```

The following is sample output from the `debug snmp packet` command. The output provides SNMP trap information for an SSH session.

```
Switch# debug snmp packet
SNMP packet debugging is on
Device1# ssh -l lab 10.0.0.2
Password:
Switch# 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.1.10.0.0.1.22.10.0.0.2.55246 = 1015
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
Switch#
```

**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 Version 2 Support

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

SUMMARY STEPS

1. enable
2. configure terminal
3. hostname name
4. ip domain-name name
5. crypto key generate rsa
6. ip ssh [time-out seconds | authentication-retries integer]
7. ip ssh version [1 | 2]
8. exit

DETAILED STEPS

<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 cisco7200</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> cisco7200(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> cisco7200(config)# crypto key generate rsa</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong>  ip ssh [time-out seconds</td>
<td>authentication-retries integer]</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
</tbody>
</table>
### Configuring a Device for SSH Version 2 Using RSA Key Pairs

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. `ip ssh rsa keypair-name keypair-name`
4. `crypto key generate rsa usage-keys label key-label modulus modulus-size`
5. `ip ssh [time-out seconds | authentication-retries integer]`
6. `ip ssh version 2`
7. `exit`

**DETAILED STEPS**

<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&gt; configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> ip ssh rsa keypair-name keypair-name</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 key-label modulus modulus-size</td>
<td>Enables the SSH server for local and remote authentication on the device.</td>
</tr>
</tbody>
</table>

(Optional) Specifies the version of SSH to be run on your device.

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>cisco7200(config)# ip ssh time-out 120</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> ip ssh version [1</td>
<td>2]</td>
</tr>
<tr>
<td>Example: cisco7200(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: cisco7200(config)# exit</td>
<td>• Use <code>no hostname</code> command to return to the default host.</td>
</tr>
</tbody>
</table>
### Configuring the Cisco SSH Server to Perform RSA-Based User Authentication

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. `hostname name`
4. `ip domain-name name`
5. `crypto key generate rsa`
6. `ip ssh pubkey-chain`
7. `username username`
8. `key-string`
9. `key-hash key-type key-name`
10. `end`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</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: Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
</tbody>
</table>

### Example:

**Purpose**

- For SSH Version 2, the modulus size must be at least 768 bits.

**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.

**Step 5** `ip ssh [time-out seconds | authentication-retries integer]`

**Example:**

Device(config)# ip ssh time-out 12

**Step 6** `ip ssh version 2`

**Example:**

Device(config)# ip ssh version 2

**Step 7** `exit`

**Example:**

Device(config)# exit

Secure Shell Version 2 Support
## Command or Action

| Step 2 | configure terminal |
|        | Example: |
|        | Device# configure terminal |

| Step 3 | hostname name |
|        | Example: |
|        | Device(config)# hostname host1 |

| Step 4 | ip domain-name name |
|        | Example: |
|        | host1(config)# ip domain-name name1 |

| Step 5 | crypto key generate rsa |
|        | Example: |
|        | host1(config)# crypto key generate rsa |

| Step 6 | ip ssh pubkey-chain |
|        | Example: |
|        | host1(config)# ip ssh pubkey-chain |

| Step 7 | username username |
|        | Example: |
|        | host1(conf-ssh-pubkey)# username user1 |

| Step 8 | key-string |
|        | Example: |
|        | host1(conf-ssh-pubkey-user)# key-string |

| Step 9 | key-hash key-type key-name |
|        | Example: |
|        | host1(conf-ssh-pubkey-data)# key-hash ssh-rsa key1 |

## Purpose

- **Step 2**: Enters global configuration mode.
- **Step 3**: Specifies the hostname.
- **Step 4**: Defines a default domain name that the Cisco software uses to complete unqualified hostnames.
- **Step 5**: Generates RSA key pairs.
- **Step 6**: Configures SSH-RSA keys for user and server authentication on the SSH server and enters public-key configuration mode.
- **Step 7**: Configures the SSH username and enters public-key user configuration mode.
- **Step 8**: Specifies the RSA public key of the remote peer and enters public-key data configuration mode.
- **Step 9**: (Optional) Specifies the SSH key type and version.

### Secure Shell Version 2 Support

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

- 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.

**Note**

You can obtain the public key value from an open SSH client; that is, from the .ssh/id_rsa.pub file.

- The key type must be ssh-rsa for the configuration of private public key pairs.
- This step is optional only if the key-string command is configured.
- You must configure either the key-string command or the key-hash command.
### Configuring the Cisco IOS SSH Client to Perform RSA-Based Server Authentication

#### SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `hostname name`
4. `ip domain-name name`
5. `crypto key generate rsa`
6. `ip ssh pubkey-chain`
7. `server server-name`
8. `key-string`
9. `exit`
10. `key-hash key-type key-name`
11. `end`
12. `configure terminal`
13. `ip ssh stricthostkeycheck`

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td></td>
</tr>
<tr>
<td><code>enable</code></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device&gt; enable</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td><code>configure terminal</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>Device# configure terminal</code></td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Step 3</strong> hostname name</td>
<td>Specifies the hostname.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device(config)# hostname host1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> ip domain-name name</td>
<td>Defines a default domain name that the Cisco software uses to complete</td>
</tr>
<tr>
<td>Example:</td>
<td>unqualified hostnames.</td>
</tr>
<tr>
<td>host1(config)# ip domain-name name1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> crypto key generate rsa</td>
<td>Generates RSA key pairs.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>host1(config)# crypto key generate rsa</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> ip ssh pubkey-chain</td>
<td>Configures SSH-RSA keys for user and server authentication on the SSH</td>
</tr>
<tr>
<td>Example:</td>
<td>server and enters public-key configuration mode.</td>
</tr>
<tr>
<td>host1(config)# ip ssh pubkey-chain</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> server server-name</td>
<td>Enables the SSH server for public-key authentication on the device and</td>
</tr>
<tr>
<td>Example:</td>
<td>enters public-key server configuration mode.</td>
</tr>
<tr>
<td>host1(conf-ssh-pubkey)# server server1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> key-string</td>
<td>Specifies the RSA public-key of the remote peer and enters public-key</td>
</tr>
<tr>
<td>Example:</td>
<td>data configuration mode.</td>
</tr>
<tr>
<td>host1(conf-ssh-pubkey-server)# key-string</td>
<td>You can obtain the public key value from an open SSH client; that is,</td>
</tr>
<tr>
<td></td>
<td>from the .ssh/id_rsa.pub file.</td>
</tr>
<tr>
<td><strong>Step 9</strong> exit</td>
<td>Exits public-key data configuration mode and enters public-key server</td>
</tr>
<tr>
<td>Example:</td>
<td>configuration mode.</td>
</tr>
<tr>
<td>host1(conf-ssh-pubkey-data)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 10</strong> key-hash key-type key-name</td>
<td>(Optional) Specifies the SSH key type and version.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>host1(conf-ssh-pubkey-server)# key-hash ssh-rsa key1</td>
<td>• The key type must be ssh-rsa for the configuration of private/public key pairs.</td>
</tr>
<tr>
<td></td>
<td>• This step is optional only if the <strong>key-string</strong> command is configured.</td>
</tr>
<tr>
<td></td>
<td>• You must configure either the <strong>key-string</strong> command or the <strong>key-hash</strong> command.</td>
</tr>
</tbody>
</table>
Starting an Encrypted Session with a Remote Device

**SUMMARY STEPS**

1. `ssh [-v {1 | 2}] | -c {aes128-ctr | aes192-ctr | aes256-ctr | aes128-cbc | aes256-cbc | 3des | aes192-cbc | aes256-cbc} | -l user-id | -l user-id:vrf-name number ip-address ip-address | -m {hmac-md5-128 | hmac-md5-96 | hmac-sha1-160 | hmac-sha1-96} | -o numberofpasswordprompts n | -p port-num] {ip-addr | hostname} [command | -vrf]

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Starts an encrypted session with a remote networking device.</td>
</tr>
</tbody>
</table>

**Note**

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.

---

Purpose

**Command or Action**

**Purpose**

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.

**Note**

**Step 11**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>end</td>
<td>Exits public-key server configuration mode and returns to privileged EXEC mode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip ssh stricthostkeycheck</td>
<td>Ensures that server authentication takes place.</td>
</tr>
</tbody>
</table>

- The connection is terminated in case of a failure.
- Use **no hostname** command to return to the default host.

---

Starting an Encrypted Session with a Remote Device

**Note**

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.

**SUMMARY STEPS**

1. `ssh [-v {1 | 2}] | -c {aes128-ctr | aes192-ctr | aes256-ctr | aes128-cbc | aes256-cbc | 3des | aes192-cbc | aes256-cbc} | -l user-id | -l user-id:vrf-name number ip-address ip-address | -m {hmac-md5-128 | hmac-md5-96 | hmac-sha1-160 | hmac-sha1-96} | -o numberofpasswordprompts n | -p port-num] {ip-addr | hostname} [command | -vrf]`
Enabling Secure Copy Protocol on the SSH Server

**Note**
The following task configures the server-side functionality for SCP. This task shows a typical configuration that allows the device to securely copy files from a remote workstation.

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. `aaa new-model`
4. `aaa authentication login default local`
5. `aaa authorization exec default local`
6. `username name privilege privilege-level password password`
7. `ip ssh time-out seconds`
8. `ip ssh authentication-retries integer`
9. `ip scpserverenable`
10. `exit`
11. `debug ip scp`

**DETAILED STEPS**

<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:</td>
<td><code>Device&gt; enable</code></td>
</tr>
<tr>
<td></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>Example:</td>
<td><code>Device# configure terminal</code></td>
</tr>
<tr>
<td><strong>Step 3</strong> aaa new-model</td>
<td>Enables the AAA access control model.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>4</td>
<td>Device(config)# aaa new-model</td>
</tr>
<tr>
<td></td>
<td><strong>Step 4</strong> aaaa authentication login default local</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config)# aaa authentication login default local</td>
</tr>
<tr>
<td>5</td>
<td>Device(config)# aaaa authorization exec default local</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config)# aaaa authorization exec default local</td>
</tr>
<tr>
<td>6</td>
<td>username sample-name privilege privilege-level password password</td>
</tr>
<tr>
<td></td>
<td>Example: Device(config)# username sample-name privilege 15 password password1</td>
</tr>
<tr>
<td>7</td>
<td>Device(config)# ip ssh time-out 120</td>
</tr>
<tr>
<td>8</td>
<td>Device(config)# ip ssh authentication-retries 3</td>
</tr>
<tr>
<td>9</td>
<td>Device(config)# ip scp server enable</td>
</tr>
<tr>
<td>10</td>
<td>Device(config)# exit</td>
</tr>
<tr>
<td>11</td>
<td>Device# debug ip scp</td>
</tr>
</tbody>
</table>
Verifying the Status of the Secure Shell Connection

SUMMARY STEPS

1. enable
2. show ssh
3. exit

DETAILED STEPS

<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:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> show ssh</td>
<td>Displays the status of SSH server connections.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# show ssh</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> exit</td>
<td>Exits privileged EXEC mode and returns to user EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# exit</td>
<td></td>
</tr>
</tbody>
</table>

**Examples**

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
```

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
0 2.0 IN aes128-cbc hmac-md5 Session started lab
1 2.0 OUT aes128-cbc hmac-md5 Session started lab
```

Secure Shell Version 2 Support
Verifying the Secure Shell Status

SUMMARY STEPS

1. enable
2. show ip ssh
3. exit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>Step 2 show ip ssh</td>
<td>Displays the version and configuration data for SSH.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# show ip ssh</td>
<td></td>
</tr>
<tr>
<td>Step 3 exit</td>
<td>Exits privileged EXEC mode and returns to user EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# exit</td>
<td></td>
</tr>
</tbody>
</table>

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

Connection | Version | Encryption | State       | Username |
-----------|---------|------------|-------------|----------|
0          | 1.5     | 3DES       | Session started | lab      |
%No SSHv2 server connections running.

-----------------------------------------------------------------------
Monitoring and Maintaining Secure Shell Version 2

**SUMMARY STEPS**

1. enable
2. debug ip ssh
3. debug snmp packet

**DETAILED STEPS**

<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> debug ip ssh</td>
<td>Enables debugging of SSH.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device# debug ip ssh</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> debug snmp packet</td>
<td>Enables debugging of every SNMP packet sent or received by the device.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device# debug snmp packet</td>
<td></td>
</tr>
</tbody>
</table>

### Example

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

```plaintext
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 2: kex: client->server aes128-cbc hmac-md5 none
00:33:55: SSH2 2: 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 2: kex_derive_keys complete
00:33:55: SSH2 1: send: len 16 (includes padlen 10)
00:33:55: SSH2 1: newkeys: mode 1
00:33:55: SSH2 1: SSH2_MSG_NEWKEYS sent
00:33:55: SSH2 1: waiting for SSH2_MSG_NEWKEYS
00:33:55: SSH2 1: ssh_receive: 16 bytes received
00:33:55: SSH2 1: input: packet len 16
00:33:55: SSH2 1: partial packet 8, need 8, maclen 0
00:33:55: SSH2 1: input: padlen 10
00:33:55: SSH2 1: newkeys: mode 0
00:33:55: SSH2 1: received packet type 2100:33:55: SSH2 1: SSH2_MSG_NEWKEYS received
00:33:55: SSH2 1: ssh_receive: 48 bytes received
00:33:56: SSH2 1: input: packet len 32
00:33:56: SSH2 1: partial packet 16, need 16, maclen 16
00:33:56: SSH2 1: MAC #3 ok
00:33:56: SSH2 1: input: padlen 10
00:33:56: SSH2 1: received packet type 5
00:33:56: SSH2 1: send: len 32 (includes padlen 10)
00:33:56: SSH2 2: done calc MAC out #3
00:33:56: SSH2 1: ssh_receive: 64 bytes received
00:33:56: SSH2 1: input: packet len 48
00:33:56: SSH2 1: partial packet 16, need 32, maclen 16
00:33:56: SSH2 1: MAC #4 ok
00:33:56: SSH2 1: input: padlen 9
00:33:56: SSH2 1: received packet type 50
```

---

Secure Shell Version 2 Support

Monitoring and Maintaining Secure Shell Version 2
00:33:56: SSH2 1: send: len 32 (includes padlen 13)
00:33:56: SSH2 1: done calc MAC out #4
00:34:04: SSH2 1: ssh_receive: 160 bytes received
00:34:04: SSH2 1: input: packet len 64
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #5 ok
00:34:04: SSH2 1: input: padlen 13
00:34:04: SSH2 1: received packet type 50
00:34:04: SSH2 1: send: len 16 (includes padlen 10)
00:34:04: SSH2 1: done calc MAC out #5
00:34:04: SSH2 1: authentication successful for lab
00:34:04: SSH2 1: input: packet len 64
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #6 ok
00:34:04: SSH2 1: input: padlen 6
00:34:04: SSH2 1: received packet type 2
00:34:04: SSH2 1: ssh_receive: 64 bytes received
00:34:04: SSH2 1: input: packet len 48
00:34:04: SSH2 1: partial packet 16, need 32, maclen 16
00:34:04: SSH2 1: MAC #7 ok
00:34:04: SSH2 1: input: padlen 19
00:34:04: SSH2 1: received packet type 90
00:34:04: SSH2 1: channel open request
00:34:04: SSH2 1: send: len 32 (includes padlen 10)
00:34:04: SSH2 1: done calc MAC out #6
00:34:04: SSH2 1: ssh_receive: 192 bytes received
00:34:04: SSH2 1: input: packet len 64
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #8 ok
00:34:04: SSH2 1: input: padlen 13
00:34:04: SSH2 1: received packet type 98
00:34:04: SSH2 1: pty-reg request
00:34:04: SSH2 1: setting TTY - requested: height 24, width 80; set: height 24, width 80
00:34:04: SSH2 1: input: packet len 96
00:34:04: SSH2 1: partial packet 16, need 80, maclen 16
00:34:04: SSH2 1: MAC #9 ok
00:34:04: SSH2 1: input: padlen 11
00:34:04: SSH2 1: received packet type 98
00:34:04: SSH2 1: x11-reg request
00:34:04: SSH2 1: ssh_receive: 48 bytes received
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
Configuration Examples for Secure Shell Version 2 Support

Example: Configuring Secure Shell Version 2

Device# configure terminal
Device(config)# ip ssh version 2

Example: Starting an Encrypted Session with a Remote Device

Device# ssh -v 2 -c aes256-cbc -m hmac-sha1-160 -l sship 10.76.82.24

Example: Configuring Server-Side SCP

The following example shows how to configure the server-side functionality for SCP. This example also configures AAA authentication and authorization on the device. This example uses a locally defined username and password.
Device# configure terminal
Device(config)# aaa new-model
Device(config)# aaa authentication login default local
Device(config)# aaa authorization exec default local
Device(config)# username samplename privilege 15 password password1
Device(config)# ip ssh time-out 120
Device(config)# ip ssh authentication-retries 3
Device(config)#

Example: Setting an SNMP Trap

The following example shows that an SNMP trap is set. The trap notification is generated automatically when the SSH session terminates. In the example, a.b.c.d is the IP address of the SSH client.

snmp-server
snmp-server host a.b.c.d public tty

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#
[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
tcpConnEntry.5.10.0.0.1.22.10.0.0.2.55246 = 1015
tcpConnEntry.1.10.0.0.1.22.10.0.0.2.55246 = 1056
tcpConnEntry.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

Examples: SSH Keyboard Interactive Authentication

Example: Enabling Client-Side Debugs

The following example shows that the client-side debugs 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:
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.

Device1# ssh -l cisco 10.1.1.3
Password:
Old Password: cisco
New Password: cisco123
Re-enter New password: cisco123
Device2> exit
[Connection to 10.1.1.3 closed by foreign host]

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# 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: cisco12
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.

Device1# 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

[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: 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.11.10.0.0.1.22.10.0.0.2.55246 = 1056
ltcpConnEntry.2.10.0.0.1.22.10.0.0.2.55246 = 1392
local.9.2.1.872 = 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.

```
Device# 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: waiting for SSH2_MSG_NEWKEYS
00:04:22: SSH2 0: SSH2_MSG_NEWKEYS received
00:04:24: SSH2 0: authentication successful for lab
00:04:24: 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.
Device# 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: 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: 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: 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: 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: 64 bytes received
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

<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>Cisco security commands</td>
<td>• Cisco IOS Security Command Reference: Commands A to C</td>
</tr>
<tr>
<td></td>
<td>• Cisco IOS Security Command Reference: Commands D to L</td>
</tr>
<tr>
<td></td>
<td>• Cisco IOS Security Command Reference: Commands M to R</td>
</tr>
<tr>
<td></td>
<td>• Cisco IOS Security Command Reference: Commands S to Z</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 and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.</td>
<td><a href="http://www.cisco.com/cisco/web/support/index.html">http://www.cisco.com/cisco/web/support/index.html</a></td>
</tr>
</tbody>
</table>

Feature Information for Secure Shell Version 2 Support

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure Shell Version 2 Client and Server Support</td>
<td>Cisco IOS XE Release 3.4SG</td>
<td>The Cisco image was updated to provide for the automatic generation of SNMP traps when an SSH session terminates. This feature was supported on CAT2960, CAT3560E, CAT3560X, CAT3750, CAT3750E, CAT3750X, CAT4500.</td>
</tr>
</tbody>
</table>
### Feature Information for Secure Shell Version 2 Support

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure Shell Version 2 Enhancements</td>
<td>Cisco IOS XE Release 3.4SG</td>
<td>The Secure Shell Version 2 Enhancements feature includes a number of additional capabilities such as support for VRF-Aware SSH, SSH debug enhancements, and DH Group 14 and Group 16 exchange support. This feature was supported on CAT2960, CAT3560E, CAT3560X, CAT3750, CAT3750E, CAT3750X, CAT4500. <strong>Note</strong> The VRF-Aware SSH feature is supported depending on your release. The following commands were introduced or modified: <code>debug ip ssh</code>, and <code>ip ssh dh min size</code>.</td>
</tr>
<tr>
<td>Secure Shell Version 2 Enhancements for RSA Keys</td>
<td>Cisco IOS XE Release 3.4SG</td>
<td>The Secure Shell Version 2 Enhancements for RSA Keys feature includes a number of additional capabilities to support RSA key-based user authentication for SSH and SSH server host key storage and verification. This feature was supported on CAT2960, CAT3560E, CAT3560X, CAT3750, CAT3750E, CAT3750X, CAT4500.</td>
</tr>
</tbody>
</table>
| Secure Shell Version 2 Support                   | Cisco IOS XE Release 3.4SG      | 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. This feature was supported on CAT2960, CAT3560E, CAT3560X, CAT3750, CAT3750E, CAT3750X, CAT4500. The following commands were introduced or modified: `debug ip ssh`, `ip ssh min dh size`, `ip ssh rsa keypair-name`, `ip ssh version`, and `ssh`.

---

**Secure Shell Version 2 Support**
### Feature Information for Secure Shell Version 2 Support

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSH Keyboard Interactive Authentication</td>
<td>Cisco IOS XE Release 3.4SG</td>
<td>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. This feature was supported on CAT2960, CAT3560E, CAT3560X, CAT3750, CAT3750E, CAT3750X, CAT4500.</td>
</tr>
</tbody>
</table>