The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

Notwithstanding any other warranty herein, all document files and software of these suppliers are provided “as is” with all faults. Cisco and the above-named suppliers disclaim all warranties, expressed or implied, including, without limitation, those of merchantability, fitness for a particular purpose and noninfringement or arising from a course of dealing, usage, or trade practice. In no event shall Cisco or its suppliers be liable for any indirect, special, consequential, or incidental damages, including, without limitation, lost profits or loss or damage to data arising out of the use or inability to use this manual, even if Cisco or its suppliers have been advised of the possibility of such damages.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2019 Cisco Systems, Inc. All rights reserved.
# Contents

## Chapter 1

**Read Me First**  
1

## Chapter 2

**Reverse SSH Enhancements**  
3

- Finding Feature Information  
3
- Prerequisites for Reverse SSH Enhancements  
3
- Restrictions for Reverse SSH Enhancements  
4
- Information About Reverse SSH Enhancements  
4
  - Reverse Telnet  
4
  - Reverse SSH  
4
- How to Configure Reverse SSH Enhancements  
4
  - Configuring Reverse SSH for Console Access  
4
  - Configuring Reverse SSH for Modem Access  
6
- Troubleshooting Reverse SSH on the Client  
8
- Troubleshooting Reverse SSH on the Server  
8
- Configuration Examples for Reverse SSH Enhancements  
9
  - Example Reverse SSH Console Access  
9
  - Example Reverse SSH Modem Access  
9
- Additional References  
10
  - Related Documents  
10
  - Technical Assistance  
10
- Standards  
11
  - MIBs  
11
  - RFCs  
11
- Technical Assistance  
11
- Feature Information for Reverse SSH Enhancements  
11
### CHAPTER 3

**Secure Copy** 13

- Prerequisites for Secure Copy 13
- Restrictions for Secure Copy Performance Improvement 13
- Information About Secure Copy 14
  - How SCP Works 14
- How to Configure SCP 14
  - Configuring SCP 14
  - Verifying SCP 15
  - Troubleshooting SCP 16
- Configuration Examples for Secure Copy 16
  - Example SCP Server-Side Configuration Using Local Authentication 16
  - Example SCP Server-Side Configuration Using Network-Based Authentication 17
- Additional References 17
- Feature Information for Secure Copy 18
- Glossary 18

### CHAPTER 4

**Secure Shell Version 2 Support** 21

- Finding Feature Information 21
- Prerequisites for Secure Shell Version 2 Support 21
- Restrictions for Secure Shell Version 2 Support 22
- Information About Secure Shell Version 2 Support 22
  - Secure Shell Version 2 22
  - Secure Shell Version 2 Enhancements 23
  - Secure Shell Version 2 Enhancements for RSA Keys 23
  - SNMP Trap Generation 24
  - SSH Keyboard Interactive Authentication 24
- How to Configure Secure Shell Version 2 Support 25
  - Configuring a Device for SSH Version 2 Using a Hostname and Domain Name 25
  - Configuring a Device for SSH Version 2 Using RSA Key Pairs 26
  - Configuring the Cisco SSH Server to Perform RSA-Based User Authentication 27
  - Configuring the Cisco IOS SSH Client to Perform RSA-Based Server Authentication 29
  - Starting an Encrypted Session with a Remote Device 31
  - Troubleshooting Tips 32
Read Me First

Important Information about Cisco IOS XE 16

Effective Cisco IOS XE Release 3.7.0E for Catalyst Switching and Cisco IOS XE Release 3.17S (for Access and Edge Routing) the two releases evolve (merge) into a single version of converged release—the Cisco IOS XE 16—providing one release covering the extensive range of access and edge products in the Switching and Routing portfolio.

Feature Information

Use Cisco Feature Navigator to find information about feature support, platform support, and Cisco software image support. An account on Cisco.com is not required.

Related References

• Cisco IOS Command References, All Releases

Obtaining Documentation and Submitting a Service Request

• To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
• To get the business impact you’re looking for with the technologies that matter, visit Cisco Services.
• To submit a service request, visit Cisco Support.
• To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco Marketplace.
• To obtain general networking, training, and certification titles, visit Cisco Press.
• To find warranty information for a specific product or product family, access Cisco Warranty Finder.
Reverse SSH Enhancements

The Reverse SSH Enhancements feature, which is supported for SSH Version 1 and 2, provides an alternative way to configure reverse Secure Shell (SSH) so that separate lines do not need to be configured for every terminal or auxiliary line on which SSH must be enabled. This feature also eliminates the rotary-group limitation.

- Finding Feature Information, on page 3
- Prerequisites for Reverse SSH Enhancements, on page 3
- Restrictions for Reverse SSH Enhancements, on page 4
- Information About Reverse SSH Enhancements, on page 4
- How to Configure Reverse SSH Enhancements, on page 4
- Configuration Examples for Reverse SSH Enhancements, on page 9
- Additional References, on page 10
- Feature Information for Reverse SSH Enhancements, on page 11

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.

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.

Prerequisites for Reverse SSH Enhancements

- SSH must be enabled.
- The SSH client and server must be running the same version of SSH.
Restrictions for Reverse SSH Enhancements

- The -I keyword and userid:{number} {ip-address} delimiter and arguments are mandatory when configuring the alternative method of Reverse SSH for console access.

Information About Reverse SSH Enhancements

Reverse Telnet

Reverse telnet allows you to telnet to a certain port range and connect to terminal or auxiliary lines. Reverse telnet has often been used to connect a Cisco device that has many terminal lines to the consoles of other Cisco devices. Telnet makes it easy to reach the device console from anywhere simply by telnet to the terminal server on a specific line. This telnet approach can be used to configure a device even if all network connectivity to that device is disconnected. Reverse telnet also allows modems that are attached to Cisco devices to be used for dial-out (usually with a rotary device).

Reverse SSH

Reverse telnet can be accomplished using SSH. Unlike reverse telnet, SSH provides for secure connections. The Reverse SSH Enhancements feature provides you with a simplified method of configuring SSH. Using this feature, you no longer have to configure a separate line for every terminal or auxiliary line on which you want to enable SSH. The previous method of configuring reverse SSH limited the number of ports that can be accessed to 100. The Reverse SSH Enhancements feature removes the port number limitation. For information on the alternative method of configuring reverse SSH, see How to Configure Reverse SSH Enhancements, on page 4.

How to Configure Reverse SSH Enhancements

Configuring Reverse SSH for Console Access

To configure reverse SSH console access on the SSH server, perform the following steps.

SUMMARY STEPS

1. enable
2. configure terminal
3. line line-number ending-line-number
4. no exec
5. login authentication listname
6. transport input ssh
7. exit
8. exit
### 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:</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>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>line line-number ending-line-number</td>
<td>Identifies a line for configuration and enters line configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device# line 1 3</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>no exec</td>
<td>Disables EXEC processing on a line.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-line)# no exec</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>login authentication listname</td>
<td>Defines a login authentication mechanism for the lines.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>Note: The authentication method must use a username and password.</td>
</tr>
<tr>
<td></td>
<td>Device(config-line)# login authentication default</td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td>transport input ssh</td>
<td>Defines which protocols to use to connect to a specific line of the device.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>• The ssh keyword must be used for the Reverse SSH Enhancements feature.</td>
</tr>
<tr>
<td></td>
<td>Device(config-line)# transport input ssh</td>
<td></td>
</tr>
<tr>
<td>Step 7</td>
<td>exit</td>
<td>Exits line configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config-line)# exit</td>
<td></td>
</tr>
<tr>
<td>Step 8</td>
<td>exit</td>
<td>Exits global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device(config)# exit</td>
<td></td>
</tr>
<tr>
<td>Step 9</td>
<td>ssh -l userid : {number} {ip-address}</td>
<td>Specifies the user ID to use when logging in on the remote networking device that is running the SSH server.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>• userid --User ID.</td>
</tr>
</tbody>
</table>
Device# ssh -l lab:1 router.example.com

• : --Signifies that a port number and terminal IP address will follow the userid argument.
  • number --Terminal or auxiliary line number.
  • ip-address --Terminal server IP address.

Note The userid argument and :rotary{number}{ip-address} delimiter and arguments are mandatory when configuring the alternative method of Reverse SSH for modem access.

Configuring Reverse SSH for Modem Access

To configure Reverse SSH for modem access, perform the steps shown in the “SUMMARY STEPS” section below.

In this configuration, reverse SSH is being configured on a modem used for dial-out lines. To get any of the dial-out modems, you can use any SSH client and start a SSH session as shown (in Step 10) to get to the next available modem from the rotary device.

SUMMARY STEPS

1. enable
2. configure terminal
3. line line-number ending-line-number
4. no exec
5. login authentication listname
6. rotary group
7. transport input ssh
8. exit
9. exit
10. ssh -l userid :rotary {number} {ip-address}

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>Example:</td>
<td>Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Step 2</td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>line line-number ending-line-number</strong>&lt;br&gt;Example:&lt;br&gt;Device# line 1 200</td>
<td>Identifies a line for configuration and enters line configuration mode.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>no exec</strong>&lt;br&gt;Example:&lt;br&gt;Device(config-line)# no exec</td>
<td>Disables EXEC processing on a line.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>login authentication listname</strong>&lt;br&gt;Example:&lt;br&gt;Device(config-line)# login authentication default</td>
<td>Defines a login authentication mechanism for the lines. Note: The authentication method must use a username and password.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>rotary group</strong>&lt;br&gt;Example:&lt;br&gt;Device(config-line)# rotary 1</td>
<td>Defines a group of lines consisting of one or more virtual terminal lines or one auxiliary port line.</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>transport input ssh</strong>&lt;br&gt;Example:&lt;br&gt;Device(config-line)# transport input ssh</td>
<td>Defines which protocols to use to connect to a specific line of the device. &lt;ul&gt;&lt;li&gt;The ssh keyword must be used for the Reverse SSH Enhancements feature.&lt;/li&gt;&lt;/ul&gt;</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td><strong>exit</strong>&lt;br&gt;Example:&lt;br&gt;Device(config-line)# exit</td>
<td>Exits line configuration mode.</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td><strong>exit</strong>&lt;br&gt;Example:&lt;br&gt;Device(config)# exit</td>
<td>Exits global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td><strong>ssh -l userid :rotary {number} {ip-address}</strong>&lt;br&gt;Example:&lt;br&gt;Device# ssh -l lab:rotary1 router.example.com</td>
<td>Specifies the user ID to use when logging in on the remote networking device that is running the SSH server. &lt;ul&gt;&lt;li&gt;<strong>userid</strong> --User ID.&lt;/li&gt;&lt;li&gt;<strong>:</strong> --Signifies that a port number and terminal IP address will follow the userid argument.&lt;/li&gt;&lt;li&gt;<strong>number</strong> --Terminal or auxiliary line number.&lt;/li&gt;&lt;li&gt;<strong>ip-address</strong> --Terminal server IP address.&lt;/li&gt;&lt;/ul&gt;</td>
</tr>
</tbody>
</table>
The \textit{userid} argument and the \texttt{:rotary\{number\}\{ip-address\}} delimiter and arguments are mandatory when configuring the alternative method of Reverse SSH for modem access.

### Troubleshooting Reverse SSH on the Client

To troubleshoot the reverse SSH configuration on the client (remote device), perform the following steps.

**SUMMARY STEPS**

1. enable
2. debug ip ssh client

**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>* Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> debug ip ssh client</td>
<td>Displays debugging messages for the SSH client.</td>
</tr>
<tr>
<td>Example: Device# debug ip ssh client</td>
<td></td>
</tr>
</tbody>
</table>

### Troubleshooting Reverse SSH on the Server

To troubleshoot the reverse SSH configuration on the terminal server, perform the following steps. The steps may be configured in any order or independent of one another.

**SUMMARY STEPS**

1. enable
2. debug ip ssh
3. show ssh
4. show line

**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>
</tbody>
</table>
### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td>• Enter your password if prompted.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td>debug ip ssh</td>
<td>Displays debugging messages for the SSH server.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device# debug ip ssh</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td></td>
</tr>
<tr>
<td>show ssh</td>
<td>Displays the status of the SSH server connections.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device# show ssh</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td>show line</td>
<td>Displays parameters of a terminal line.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device# show line</td>
<td></td>
</tr>
</tbody>
</table>

### Configuration Examples for Reverse SSH Enhancements

#### Example Reverse SSH Console Access

The following configuration example shows that reverse SSH has been configured for console access for terminal lines 1 through 3:

**Terminal Server Configuration**

```
line 1 3
   no exec
   login authentication default
   transport input ssh
```

**Client Configuration**

The following commands configured on the SSH client will form the reverse SSH session with lines 1, 2, and 3, respectively:

```
ssh -l lab:1 router.example.com
ssh -l lab:2 router.example.com
ssh -l lab:3 router.example.com
```

#### Example Reverse SSH Modem Access

The following configuration example shows that dial-out lines 1 through 200 have been grouped under rotary group 1 for modem access:

---

Secure Shell Configuration Guide, Cisco IOS XE Gibraltar 16.11.x
line 1 200
  no exec
  login authentication default
  rotary 1
  transport input ssh
  exit

The following command shows that reverse SSH will connect to the first free line in the rotary group:

```bash
ssh -l lab:rotary1 router.example.com
```

### Additional References

#### 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 Commands List, All Releases</td>
</tr>
<tr>
<td>Configuring Secure Shell</td>
<td>Secure Shell Configuration Guide</td>
</tr>
<tr>
<td>Security commands</td>
<td>Cisco IOS Security Command Reference</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</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>
<tr>
<td>install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the</td>
<td></td>
</tr>
<tr>
<td>Cisco Support and Documentation website requires a Cisco.com user ID and password.</td>
<td></td>
</tr>
</tbody>
</table>

#### 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 Commands List, All Releases</td>
</tr>
<tr>
<td>Configuring Secure Shell</td>
<td>Secure Shell Configuration Guide</td>
</tr>
<tr>
<td>Security commands</td>
<td>Cisco IOS Security Command Reference</td>
</tr>
</tbody>
</table>
Standards

<table>
<thead>
<tr>
<th>Standards</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new or modified standards are supported by this feature.</td>
<td>--</td>
</tr>
</tbody>
</table>

MIBs

<table>
<thead>
<tr>
<th>MIBs</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: &lt;br&gt;<a href="http://www.cisco.com/go/mibs">http://www.cisco.com/go/mibs</a></td>
</tr>
</tbody>
</table>

RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>--</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 Reverse SSH Enhancements

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 1: Feature Information for Reverse SSH Enhancements

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reverse SSH Enhancements</td>
<td></td>
<td>The Reverse SSH Enhancements feature, which is supported for SSH Version 1 and 2, provides an alternative way to configure reverse Secure Shell (SSH) so that separate lines do not need to be configured for every terminal or auxiliary line on which SSH must be enabled. This feature also eliminates the rotary-group limitation. The following command was introduced: <code>ssh</code>.</td>
</tr>
</tbody>
</table>
CHAPTER 3

Secure Copy

The Secure Copy Protocol (SCP) feature provides a secure and authenticated method for copying router configuration or router image files. SCP relies on Secure Shell (SSH), an application and a protocol that provide a secure replacement for the Berkeley r-tools.

- Prerequisites for Secure Copy, on page 13
- Restrictions for Secure Copy Performance Improvement, on page 13
- Information About Secure Copy, on page 14
- How to Configure SCP, on page 14
- Configuration Examples for Secure Copy, on page 16
- Additional References, on page 17
- Feature Information for Secure Copy, on page 18
- Glossary, on page 18

Prerequisites for Secure Copy

- Before enabling SCP, you must correctly configure SSH, authentication, and authorization on the router.
- Because SCP relies on SSH for its secure transport, the router must have an Rivest, Shamir, and Adelman (RSA) key pair.

Restrictions for Secure Copy Performance Improvement

- Incrementing window-size must be used mainly for SCP operations only.
- Depending on the platform type, the maximum window size can cause high CPU usage.
- As a precaution, increments can be made up to four times the default size.
## Information About Secure Copy

### How SCP Works

The behavior of SCP is similar to that of remote copy (rcp), which comes from the Berkeley r-tools suite, except that SCP relies on SSH for security. In addition, SCP requires that authentication, authorization, and accounting (AAA) authorization be configured so the router can determine whether the user has the correct privilege level.

SCP allows a user who has appropriate authorization to copy any file that exists in the Cisco IOS XE File System (IFS) to and from a router by using the `copy` command. An authorized administrator may also perform this action from a workstation.

### How to Configure SCP

#### Configuring SCP

To enable and configure a Cisco router for SCP server-side functionality, perform the following steps.

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. `aaa new-model`
4. `aaa authentication login {default | list-name} method1[method2...]`
5. `aaa authorization {network | exec | commands level | reverse-access | configuration} {default | list-name} [method1 [method2...]]`
6. `username name [privilege level]{password encryption-type encrypted-password}`
7. `ip scp server enable`

#### 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>Router&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:</td>
<td></td>
</tr>
<tr>
<td>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>3</td>
<td><code>aaa new-model</code></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Router (config)# <code>aaa new-model</code></td>
</tr>
<tr>
<td>4</td>
<td>`aaa authentication login {default</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Router (config)# <code>aaa authentication login default group tacacs+</code></td>
</tr>
<tr>
<td>5</td>
<td>`aaa authorization {network</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> The <code>exec</code> keyword runs authorization to determine if the user is allowed to run an EXEC shell; therefore, you must use it when you configure SCP.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Router (config)# <code>aaa authorization exec default group tacacs+</code></td>
</tr>
<tr>
<td>6</td>
<td><code>username name [privilege level] {password encryption-type encrypted-password}</code></td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> You may skip this step if a network-based authentication mechanism--such as TACACS+ or RADIUS--has been configured.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Router (config)# <code>username superuser privilege 2 password 0 superpassword</code></td>
</tr>
<tr>
<td>7</td>
<td><code>ip scp server enable</code></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Router (config)# <code>ip scp server enable</code></td>
</tr>
</tbody>
</table>

**Verifying SCP**

To verify SCP server-side functionality, perform the following steps.

**SUMMARY STEPS**

1. `enable`
2. `show running-config`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>enable</code></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Enter your password if prompted.</td>
<td></td>
</tr>
</tbody>
</table>
## Troubleshooting SCP

### SUMMARY STEPS

1. `enable`
2. `debug ip scp`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1**
* enable
  * Example:
    * `Router> enable`
| Enables privileged EXEC mode.
  * Enter your password if prompted. |
| **Step 2**
* debug ip scp
  * Example:
    * `Router# debug ip scp`
| Troubleshoots SCP authentication problems. |

### Configuration Examples for Secure Copy

**Example SCP Server-Side Configuration Using Local Authentication**

The following example shows how to configure the server-side functionality of SCP. This example uses a locally defined username and password.

```
! AAA authentication and authorization must be configured properly for SCP to work.
aaa new-model
aaa authentication login default local
aaa authorization exec default local
username tiger privilege 15 password 0 lab
! SSH must be configured and functioning properly.
ip ssh time-out 120
ip ssh authentication-retries 3
ip scp server enable
```
Example SCP Server-Side Configuration Using Network-Based Authentication

The following example shows how to configure the server-side functionality of SCP using a network-based authentication mechanism:

! AAA authentication and authorization must be configured properly for SCP to work.
aaa new-model
aaa authentication login default group tacacs+
aaa authorization exec default group tacacs+
! SSH must be configured and functioning properly.
ip ssh time-out 120
ip ssh authentication-retries 3
ip scp server enable

Additional References

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 Commands List, All Releases</td>
</tr>
<tr>
<td>Security commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples</td>
<td>Cisco IOS Security Command Reference</td>
</tr>
<tr>
<td>Secure Shell</td>
<td>Configuring Secure Shell and Secure Shell Version 2 Support feature modules.</td>
</tr>
<tr>
<td>Configuring authentication and authorization</td>
<td>Configuring Authentication , Configuring Authorization , and Configuring Accounting feature modules.</td>
</tr>
</tbody>
</table>

Standards

<table>
<thead>
<tr>
<th>Standards</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>--</td>
</tr>
</tbody>
</table>

MIBs

<table>
<thead>
<tr>
<th>MIBs</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: <a href="http://www.cisco.com/go/mibs">http://www.cisco.com/go/mibs</a></td>
</tr>
</tbody>
</table>
RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new or modified RFCs are supported by this feature, and support</td>
<td>--</td>
</tr>
<tr>
<td>for existing RFCs has not been modified by this feature.</td>
<td></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</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>
<tr>
<td>resources to download documentation, software, and tools. Use</td>
<td></td>
</tr>
<tr>
<td>these resources to install and configure the software and to</td>
<td></td>
</tr>
<tr>
<td>troubleshoot and resolve technical issues with Cisco products</td>
<td></td>
</tr>
<tr>
<td>and technologies. Access to most tools on the Cisco Support and</td>
<td></td>
</tr>
<tr>
<td>Documentation website requires a Cisco.com user ID and password.</td>
<td></td>
</tr>
</tbody>
</table>

Feature Information for Secure Copy

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 2: Feature Information for Secure Copy

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Configuration Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure Copy</td>
<td>Cisco IOS XE</td>
<td>The Secure Copy (SCP) feature provides a secure and authenticated method for copying router</td>
</tr>
<tr>
<td></td>
<td>Release 2.1</td>
<td>configuration or router image files. SCP relies on Secure Shell (SSH), an application and a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>protocol that provide a secure replacement for the Berkeley r-tools.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In Cisco IOS XE Release 2.1, this feature was introduced on Cisco ASR 1000 Series Aggregation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Services Routers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The following commands were introduced or modified: debug ip scp, ip scp server enable.</td>
</tr>
</tbody>
</table>

Glossary

AAA --authentication, authorization, and accounting. Framework of security services that provide the method for identifying users (authentication), for remote access control (authorization), and for collecting and sending security server information used for billing, auditing, and reporting (accounting).
**rcp** --remote copy. Relying on Remote Shell (Berkeley r-tools suite) for security, rcp copies files, such as router images and startup configurations, to and from routers.

**SCP** --secure copy. Relying on SSH for security, SCP support allows the secure and authenticated copying of anything that exists in the Cisco IOS XE File Systems. SCP is derived from rcp.

**SSH** --Secure Shell. Application and a protocol that provide a secure replacement for the Berkeley r-tools. The protocol secures the sessions using standard cryptographic mechanisms, and the application can be used similarly to the Berkeley rexec and rsh tools. SSH Version 1 is implemented in the Cisco IOS XE software.
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 21
- Prerequisites for Secure Shell Version 2 Support, on page 21
- Restrictions for Secure Shell Version 2 Support, on page 22
- Information About Secure Shell Version 2 Support, on page 22
- How to Configure Secure Shell Version 2 Support, on page 25
- Configuration Examples for Secure Shell Version 2 Support, on page 39
- Additional References for Secure Shell Version 2 Support, on page 44
- Feature Information for Secure Shell Version 2 Support, on page 45

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.

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.

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.
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 both k8 and k9 images depending on your release.)

For more information about downloading a software image, refer to the Configuration Fundamentals Configuration Guide.

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.

Note

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.

Note
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.

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.

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)

For examples of various scenarios in which the SSH Keyboard Interactive Authentication feature has been automatically enabled, see the “Examples: SSH Keyboard Interactive Authentication, on page 40” section.

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>
</table>
| **Step 1** enable | Enables privileged EXEC mode.  
Example:  
Device> enable |
| **Step 2** configure terminal | Enters global configuration mode.  
Example:  
Device# configure terminal |
| **Step 3** hostname name | Configures a hostname for your device.  
Example:  
Device(config)# hostname cisco7200 |
| **Step 4** ip domain-name name | Configures a domain name for your device.  
Example:  
cisco7200(config)# ip domain-name example.com |
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<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>Example: <code>crypto key generate rsa</code></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>Example: <code>ip ssh time-out 120</code></td>
</tr>
<tr>
<td><strong>Step 7</strong> ip ssh version [1</td>
<td>(Optional) Specifies the version of SSH to be run on your device.</td>
</tr>
<tr>
<td>2]</td>
<td>Example: <code>ip ssh version 1</code></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: <code>exit</code></td>
<td>• Use <code>no hostname</code> command to return to the default host.</td>
</tr>
</tbody>
</table>

### Configuring a Device for SSH Version 2 Using RSA Key Pairs

**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** ip ssh rsa keypair-name *keypair-name*

Example:
```
Device(config)# ip ssh rsa keypair-name sshkeys
```
Specifies the RSA key pair to be used for SSH.

**Note** A Cisco device can have many RSA key pairs.
Secure Shell Version 2 Support

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

Step 4  crypto key generate rsa usage-keys label key-label modulus modulus-size

Example:

Device(config)# crypto key generate rsa usage-keys label sshkeys modulus 768

Enables the SSH server for local and remote authentication on the device.

• 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

Configures SSH control variables on your device.

Step 6  ip ssh version 2

Example:

Device(config)# ip ssh version 2

Specifies the version of SSH to be run on the device.

Step 7  exit

Example:

Device(config)# exit

Exits global configuration mode and enters privileged EXEC mode.

---

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

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.

**Step 6**  
**ip ssh pubkey-chain**  
**Example:**

host1(config)# ip ssh pubkey-chain  
Configures SSH-RSA keys for user and server authentication on the SSH server and enters public-key configuration mode.

- 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.

**Step 7**  
**username** *username*  
**Example:**

host1(conf-ssh-pubkey)# username user1  
Configures the SSH username and enters public-key user configuration mode.

**Step 8**  
**key-string**  
**Example:**

host1(conf-ssh-pubkey-user)# key-string  
Specifies the RSA public key of the remote peer and enters public-key data configuration mode.

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

**Step 9**  
**key-hash** *key-type* *key-name*  
**Example:**
host1(conf-ssh-pubkey-data)# key-hash ssh-rsa key1
(Optional) Specifies the SSH key type and version.
  • 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.

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.

Step 10 end
Example:

host1(conf-ssh-pubkey-data)# end
Exits public-key data configuration mode and returns to privileged EXEC mode.
  • Use no hostname command to return to the default host.

Configuring the Cisco IOS SSH Client to Perform RSA-Based Server Authentication

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.

Step 6  ip ssh pubkey-chain
Example:

host1(config)# ip ssh pubkey-chain
Configures SSH-RSA keys for user and server authentication on the SSH server and enters public-key configuration mode.

Step 7  server  server-name
Example:

host1(conf-ssh-pubkey)# server server1
Enables the SSH server for public-key authentication on the device and enters public-key server configuration mode.

Step 8  key-string
Example:

host1(conf-ssh-pubkey-server)# key-string
Specifies the RSA public-key of the remote peer and enters public key data configuration mode.

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

Step 9  exit
Example:

host1(conf-ssh-pubkey-data)# exit
Exits public-key data configuration mode and enters public-key server configuration mode.

Step 10  key-hash  key-type  key-name
Example:

host1(conf-ssh-pubkey-server)# key-hash ssh-rsa key1
(Optional) Specifies the SSH key type and version.

  • 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.

**Note** 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.

**Step 11**

**end**

**Example:**

```bash
host1(conf-ssh-pubkey-server)# end
```

Exits public-key server configuration mode and returns to privileged EXEC mode.

**Step 12**

**configure terminal**

**Example:**

```bash
host1# configure terminal
```

Enters global configuration mode.

**Step 13**

**ip ssh stricthostkeycheck**

**Example:**

```bash
host1(config)# ip ssh stricthostkeycheck
```

Ensures that server authentication takes place.

• 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

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.

**Note**

```bash
ssh [-v {1 | 2}] [-c {aes128-ctr | aes192-ctr | aes256-ctr | aes128-cbc | aes192-cbc | aes256-cbc}] [-l user-id] [-l user-id:vrft-name] number ip-address [ip-address] [-l user-id:rotary] number 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]
```

**Example:**
Troubleshooting Tips

The `ip ssh version` command can be used for troubleshooting your SSH configuration. By changing versions, you can determine the SSH version that has a problem.

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.

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  aaa new-model

Example:

Device(config)# aaa new-model

Enables the AAA access control model.

Step 4  aaa authentication login default local

Example:

Device(config)# aaa authentication login default local

Sets AAA authentication at login to use the local username database for authentication.

Step 5  aaa authorization exec defaultlocal

Example:
Device(config)# aaa authorization exec default local

Sets the parameters that restrict user access to a network, runs the authorization to determine if the user ID is allowed to run an EXEC shell, and specifies that the system must use the local database for authorization.

**Step 6**

`username name privilege privilege-level password password`

**Example:**

Device(config)# username samplename privilege 15 password password1

Establishes a username-based authentication system, and specifies the username, privilege level, and an unencrypted password.

**Note** The minimum value for the `privilege-level` argument is 15. A privilege level of less than 15 results in the connection closing.

**Step 7**

`ip ssh time-out seconds`

**Example:**

Device(config)# ip ssh time-out 120

Sets the time interval (in seconds) that the device waits for the SSH client to respond.

**Step 8**

`ip ssh authentication-retries integer`

**Example:**

Device(config)# ip ssh authentication-retries 3

Sets the number of authentication attempts after which the interface is reset.

**Step 9**

`ip scp server enable`

**Example:**

Device(config)# ip scp server enable

Enables the device to securely copy files from a remote workstation.

**Step 10**

`exit`

**Example:**

Device(config)# exit

Exits global configuration mode and returns to privileged EXEC mode.

**Step 11**

`debug ip scp`

**Example:**

Device# debug ip scp

(Optional) Provides diagnostic information about SCP authentication problems.
Verifying the Status of the Secure Shell Connection

Step 1  enable
Example:
Device> enable
Enables privileged EXEC mode.
  • Enter your password if prompted.

Step 2  show ssh
Example:
Device# show ssh
Displays the status of SSH server connections.

Step 3  exit
Example:
Device# exit
Exits privileged EXEC mode and returns to user EXEC mode.

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
Connection  Version  Mode  Encryption  Hmac  State  Username
  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
  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.
-------------------------------------------------------------------------

Verifying the Secure Shell Status

Step 1  enable
Example:

Device> enable

Enables privileged EXEC mode.
  • Enter your password if prompted.

Step 2  show ip ssh
Example:

Device# show ip ssh
Displays the version and configuration data for SSH.

Step 3  exit
Example:

Device# exit
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

### Step 1  enable

**Example:**

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

### Step 2  debug ip ssh

**Example:**

```
Device# debug ip ssh
```

Enables debugging of SSH.

### Step 3  debug snmp packet

**Example:**

```
Device# debug snmp packet
```

Enables debugging of every SNMP packet sent or received by the device.
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 2: kex: client->server aea128-cbc hmac-md5 none
00:33:55: SSH2 2: kex: server->client aea128-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)
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
00:33:56: 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: ssh_receive: 32 bytes received
00:33:56: SSH2 1: 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
00:33:56: SSH2 1: ssh_receive: 32 bytes received
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: input: packet len 96
00:34:04: SSH2 1: partial packet 16, need 80, 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
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
### Configuration Examples for Secure Shell Version 2 Support

**Example: Configuring Secure Shell Version 1**

```
Device# configure terminal
Device(config)# ip ssh version 1
Device(config)# ip ssh version 2
```

**Example: Configuring Secure Shell Version 2**

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

**Example: Configuring Secure Shell Versions 1 and 2**

```
Device# configure terminal
Device(config)# no ip ssh version
```
Example: Starting an Encrypted Session with a Remote Device

Device# ssh -v 2 -c aes256-cbc -m hmac-shal-160 -l shaship 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)# ip scp server enable

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. For an example of SNMP trap debug output, see the “Example: SNMP Debugging, on page 42” section.

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

Examples: SSH Keyboard Interactive Authentication

Example: Enabling Client-Side Debs

The following example shows that the client-side debs 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
user1@courier:~> exit
logout
[Connection to 10.76.248.200 closed by foreign host]
Device1# debug ip ssh client

SSH Client debugging is on
Device1# ssh -l lab 10.1.1.3
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

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
[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

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.

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-reg 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
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>AAA</td>
<td>Security Configuration Guide: Securing User Services</td>
</tr>
<tr>
<td>Hostname and host domain configuration tasks</td>
<td></td>
</tr>
<tr>
<td>Secure shell configuration tasks</td>
<td>Configuration Fundamentals Configuration Guide</td>
</tr>
<tr>
<td>Downloading a software image</td>
<td></td>
</tr>
<tr>
<td>Configuration fundamentals</td>
<td></td>
</tr>
<tr>
<td>IPsec configuration tasks</td>
<td>Security Configuration Guide: Secure Connectivity</td>
</tr>
<tr>
<td>SNMP traps configuration tasks</td>
<td>SNMP 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 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 3: 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 Support</td>
<td></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. The following commands were introduced or modified: debug ip ssh, ip ssh min dh size, ip ssh rsa keypair-name, ip ssh version, ssh.</td>
</tr>
<tr>
<td>Secure Shell Version 2 Client and Server Support</td>
<td></td>
<td>The Cisco IOS image was updated to provide for the automatic generation of SNMP traps when an SSH session terminates.</td>
</tr>
<tr>
<td>SSH Keyboard Interactive Authentication</td>
<td></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.</td>
</tr>
<tr>
<td>Secure Shell Version 2 Enhancements</td>
<td></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. The following commands were introduced or modified: debug ip ssh, ip ssh dh min size.</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 for RSA Keys</td>
<td></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. The following commands were introduced or modified: <code>ip ssh pubkey-chain</code>, <code>ip ssh stricthostkeycheck</code>.</td>
</tr>
</tbody>
</table>
Secure Shell—Configuring User Authentication Methods

The Secure Shell—Configuring User Authentication Methods feature helps configure the user authentication methods available in the Secure Shell (SSH) server.

- Finding Feature Information, on page 47
- Restrictions for Secure Shell—Configuring User Authentication Methods, on page 47
- Information About Secure Shell—Configuring User Authentication Methods, on page 48
- How to Configure Secure Shell—Configuring User Authentication Methods, on page 48
- Configuration Examples for Secure Shell—Configuring User Authentication Methods, on page 51
- Additional References for Secure Shell—Configuring User Authentication Methods, on page 52
- Feature Information for Secure Shell—Configuring User Authentication Methods, on page 53

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.

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.

Restrictions for Secure Shell—Configuring User Authentication Methods

Secure Shell (SSH) server and SSH client are supported on data encryption software (DES) (56-bit) and 3DES (168-bit) images only.
Information About Secure Shell—Configuring User Authentication Methods

Secure Shell User Authentication Overview

Secure Shell (SSH) enables an SSH client to make a secure, encrypted connection to a Cisco device (Cisco IOS SSH server). The SSH client uses the SSH protocol to provide device authentication and encryption.

The SSH server supports three types of user authentication methods and sends these authentication methods to the SSH client in the following predefined order:

- Public-key authentication method
- Keyboard-interactive authentication method
- Password authentication method

By default, all the user authentication methods are enabled. Use the `no ip ssh server authenticate user {publickey | keyboard | password}` command to disable any specific user authentication method so that the disabled method is not negotiated in the SSH user authentication protocol. This feature helps the SSH server offer any preferred user authentication method in an order different from the predefined order. The disabled user authentication method can be enabled using the `ip ssh server authenticate user {publickey | keyboard | password}` command.

As per RFC 4252 (The Secure Shell (SSH) Authentication Protocol), the public-key authentication method is mandatory. This feature enables the SSH server to override the RFC behavior and disable any SSH user authentication method, including public-key authentication.

For example, if the SSH server prefers the password authentication method, the SSH server can disable the public-key and keyboard-interactive authentication methods.

How to Configure Secure Shell—Configuring User Authentication Methods

Configuring User Authentication for the SSH Server

Perform this task to configure user authentication methods in the Secure Shell (SSH) server.

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. `no ip ssh server authenticate user {publickey | keyboard | password}`
4. `ip ssh server authenticate user {publickey | keyboard | password}`
5. `default ip ssh server authenticate user`
6. `end`
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** | `enable` | Enables privileged EXEC mode.  
  **Example:**  
  Device> enable |
| **Step 2** | `configure terminal` | Enters global configuration mode.  
  **Example:**  
  Device# configure terminal |
| **Step 3** | `no ip ssh server authenticate user {publickey | keyboard | password}` | Disables a user authentication method in the Secure Shell (SSH) server.  
  **Note** A warning message is displayed when the `no ip ssh server authenticate user publickey` command is used to disable public-key authentication. This command overrides the RFC 4252 (The Secure Shell (SSH) Authentication Protocol) behavior, which states that public-key authentication is mandatory.  
  **Example:**  
  Device(config)# no ip ssh server authenticate user publickey  
  %SSH:Publickey disabled.Overriding RFC |
| **Step 4** | `ip ssh server authenticate user {publickey | keyboard | password}` | Enables the disabled user authentication method in the SSH server.  
  **Example:**  
  Device(config)# ip ssh server authenticate user publickey |
| **Step 5** | `default ip ssh server authenticate user` | Returns to the default behavior in which all user authentication methods are enabled in the predefined order.  
  **Example:**  
  Device(config)# default ip ssh server authenticate user |
| **Step 6** | `end` | Exits global configuration mode and returns to privileged EXEC mode.  
  **Example:**  
  Device(config)# end |

### Troubleshooting Tips

- If the public-key-based authentication method is disabled using the `no ip ssh server authenticate user publickey` command, the RFC 4252 (The Secure Shell (SSH) Authentication Protocol) behavior in which public-key authentication is mandatory is overridden and the following warning message is displayed:

  %SSH:Publickey disabled.Overriding RFC

- If all three authentication methods are disabled, the following warning message is displayed:
%SSH: No auth method configured. Incoming connection will be dropped

- In the event of an incoming SSH session request from the SSH client when all three user authentication methods are disabled on the SSH server, the connection request is dropped at the SSH server and a system log message is available in the following format:

%SSH-3-NO_USERAUTH: No auth method configured for SSH Server. Incoming connection from <ip address> (tty - <ttynum>) dropped

Verifying User Authentication for the SSH Server

**SUMMARY STEPS**

1. enable
2. show ip ssh

**DETAILED STEPS**

**Step 1** enable

Enables privileged EXEC mode.

- Enter your password if prompted.

**Example:**

Device> enable

**Step 2** show ip ssh

Displays the version and configuration data for Secure Shell (SSH).

**Example:**

The following sample output from the `show ip ssh` command confirms that all three user authentication methods are enabled in the SSH server:

Device# show ip ssh

Authentication methods:publickey, keyboard-interactive, password

The following sample output from the `show ip ssh` command confirms that all three user authentication methods are disabled in the SSH server:

Device# show ip ssh

Authentication methods:NONE
Configuration Examples for Secure Shell—Configuring User Authentication Methods

Example: Disabling User Authentication Methods

The following example shows how to disable the public-key-based authentication and keyboard-based authentication methods, allowing the SSH client to connect to the SSH server using the password-based authentication method:

Device> enable
Device# configure terminal
Device(config)# no ip ssh server authenticate user publickey
%SSH:Publickey disabled.Overriding RFC
Device(config)# no ip ssh server authenticate user keyboard
Device(config)# exit

Example: Enabling User Authentication Methods

The following example shows how to enable the public-key-based authentication and keyboard-based authentication methods:

Device> enable
Device# configure terminal
Device(config)# ip ssh server authenticate user publickey
Device(config)# ip ssh server authenticate user keyboard
Device(config)# exit

Example: Configuring Default User Authentication Methods

The following example shows how to return to the default behavior in which all three user authentication methods are enabled in the predefined order:

Device> enable
Device# configure terminal
Device(config)# default ip ssh server authenticate user
Device(config)# exit
## Additional References for Secure Shell—Configuring User Authentication Methods

### 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>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>
<tr>
<td>SSH configuration</td>
<td>Secure Shell Configuration Guide</td>
</tr>
</tbody>
</table>

### Standards and RFCs

<table>
<thead>
<tr>
<th>Standard/RFC</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 4252</td>
<td>The Secure Shell (SSH) Authentication Protocol</td>
</tr>
<tr>
<td>RFC 4253</td>
<td>The Secure Shell (SSH) Transport Layer Protocol</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 documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</td>
<td><a href="http://www.cisco.com/support">http://www.cisco.com/support</a></td>
</tr>
</tbody>
</table>
Feature Information for Secure Shell—Configuring User Authentication Methods

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 4: Feature Information for Secure Shell—Configuring User Authentication Methods

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure Shell—Configuring User Authentication Methods</td>
<td>Cisco IOS XE Release 3.10S</td>
<td>The Secure Shell—Configuring User Authentication Methods feature helps configure the user authentication methods available in the Secure Shell (SSH) server. The following command was introduced: <code>ip ssh server authenticate user</code>. In Cisco IOS XE Release 3.10, this feature was introduced on Cisco ASR 1000 Series Aggregation Services Routers.</td>
</tr>
</tbody>
</table>
CHAPTER 6

X.509v3 Certificates for SSH Authentication

The X.509v3 Certificates for SSH Authentication feature uses the X.509v3 digital certificates in server and user authentication at the secure shell (SSH) server side.

This module describes how to configure server and user certificate profiles for a digital certificate.

- Finding Feature Information, on page 55
- Prerequisites for X.509v3 Certificates for SSH Authentication, on page 55
- Restrictions for X.509v3 Certificates for SSH Authentication, on page 56
- Information About X.509v3 Certificates for SSH Authentication, on page 56
- How to Configure X.509v3 Certificates for SSH Authentication, on page 56
- Configuration Examples for X.509v3 Certificates for SSH Authentication, on page 61
- Additional References for X.509v3 Certificates for SSH Authentication, on page 61
- Feature Information for X.509v3 Certificates for SSH Authentication, on page 62

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.

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.

Prerequisites for X.509v3 Certificates for SSH Authentication

- The X.509v3 Certificates for SSH Authentication feature introduces the `ip ssh server algorithm authentication` command to replace the `ip ssh server authenticate user` command. If you use the `ip ssh server authenticate user` command, the following deprecation message is displayed.

  Warning: SSH command accepted but this CLI will be deprecated soon. Please move to new CLI "ip ssh server algorithm authentication". Please configure “default ip ssh server authenticate user” to make CLI ineffective.

- Use the `default ip ssh server authenticate user` command to remove the `ip ssh server authenticate user` command from effect. The IOS secure shell (SSH) server then starts using the `ip ssh server algorithm authentication` command.
Restrictions for X.509v3 Certificates for SSH Authentication

- The X.509v3 Certificates for SSH Authentication feature implementation is applicable only on the IOS secure shell (SSH) server side.
- IOS SSH server supports only the x509v3-ssh-rsa algorithm based certificate for server and user authentication on the IOS SSH server side.

Information About X.509v3 Certificates for SSH Authentication

Digital certificates

The validity of the authentication depends upon the strength of the linkage between the public signing key and the identity of the signer. Digital certificates in the X.509v3 format (RFC5280) are used to provide identity management. A chain of signatures by a trusted root certification authority and its intermediate certificate authorities binds a given public signing key to a given digital identity.

Public key infrastructure (PKI) trustpoint helps manage the digital certificates. The association between the certificate and the trustpoint helps track the certificate. The trustpoint contains information about the certificate authority (CA), different identity parameters, and the digital certificate. Multiple trustpoints can be created to associate with different certificates.

Server and user authentication using X.509v3

For server authentication, the IOS secure shell (SSH) server sends its own certificate to the SSH client for verification. This server certificate is associated with the trustpoint configured in the server certificate profile (ssh-server-cert-profile-server configuration mode).

For user authentication, the SSH client sends the user's certificate to the IOS SSH server for verification. The SSH server validates the incoming user certificate using public key infrastructure (PKI) trustpoints configured in the server certificate profile (ssh-server-cert-profile-user configuration mode).

By default, certificate-based authentication is enabled for server and user at the IOS SSH server end.

How to Configure X.509v3 Certificates for SSH Authentication

Configuring IOS SSH Server to Use Digital Certificates for Server Authentication

SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `ip ssh server algorithm hostkey x509v3-ssh-rsa [ssh-rsa] | ssh-rsa [x509v3-ssh-rsa]`
4. `ip ssh server certificate profile`
5. server
6. trustpoint sign PKI-trustpoint-name
7. ocsp-response include
8. end

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>enable</td>
</tr>
<tr>
<td>Example:</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>configure terminal</td>
</tr>
<tr>
<td>Example:</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>ip ssh server algorithm hostkey {x509v3-ssh-rsa [ssh-rsa] [ssh-rsa [x509v3-ssh-rsa]]}</td>
</tr>
<tr>
<td>Example:</td>
<td>Defines the order of host key algorithms. Only the configured algorithm is negotiated with the secure shell (SSH) client.</td>
</tr>
<tr>
<td></td>
<td>Note: The IOS SSH server must have at least one configured host key algorithm:</td>
</tr>
<tr>
<td></td>
<td>• ssh-rsa – public key based authentication</td>
</tr>
<tr>
<td></td>
<td>• x509v3-ssh-rsa – certificate-based authentication</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>ip ssh server certificate profile</td>
</tr>
<tr>
<td>Example:</td>
<td>Configures server certificate profile and user certificate profile and enters SSH certificate profile configuration mode.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>server</td>
</tr>
<tr>
<td>Example:</td>
<td>Configures server certificate profile and enters SSH server certificate profile server configuration mode.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>trustpoint sign PKI-trustpoint-name</td>
</tr>
<tr>
<td>Example:</td>
<td>Attaches the public key infrastructure (PKI) trustpoint to the server certificate profile. The SSH server uses the certificate associated with this PKI trustpoint for server authentication.</td>
</tr>
</tbody>
</table>
Configuring IOS SSH Server to Verify User’s Digital Certificate for User Authentication

SUMMARY STEPS

1. enable
2. configure terminal
3. ip ssh server algorithm authentication {publickey | keyboard | password}
4. ip ssh server algorithm publickey {x509v3-ssh-rsa [ssh-rsa] | ssh-rsa [x509v3-ssh-rsa]}
5. ip ssh server certificate profile
6. user
7. trustpoint verify PKI-trustpoint-name
8. ocsp-response required
9. end

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> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> ip ssh server algorithm authentication {publickey</td>
<td>keyboard</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
</tbody>
</table>
### Purpose

- **Command or Action**

  Device(config)# ip ssh server algorithm authentication publickey

  **Note**  The IOS SSH server must have at least one configured user authentication algorithm.

  **Note**  To use the certificate method for user authentication, the `publickey` keyword must be configured.

  **Note**  The `ip ssh server algorithm authentication` command replaces the `ip ssh server authenticate user` command.

- **Step 4**

  `ip ssh server algorithm publickey {x509v3-ssh-rsa [ssh-rsa] | ssh-rsa [x509v3-ssh-rsa]}`

  **Example:**
  
  Device(config)# ip ssh server algorithm publickey x509v3-ssh-rsa

  Defines the order of public key algorithms. Only the configured algorithm is accepted by the SSH client for user authentication.

  **Note**  The IOS SSH client must have at least one configured public key algorithm:

  - `ssh-rsa` – public-key-based authentication
  - `x509v3-ssh-rsa` – certificate-based authentication

- **Step 5**

  `ip ssh server certificate profile`

  **Example:**
  
  Device(config)# ip ssh server certificate profile

  Configures server certificate profile and user certificate profile and enters SSH certificate profile configuration mode.

- **Step 6**

  `user`

  **Example:**
  
  Device(ssh-server-cert-profile)# user

  Configures user certificate profile and enters SSH server certificate profile user configuration mode.

- **Step 7**

  `trustpoint verify PKI-trustpoint-name`

  **Example:**
  
  Device(ssh-server-cert-profile-user)# trustpoint verify trust2

  Configures the public key infrastructure (PKI) trustpoint that is used to verify the incoming user certificate.

  **Note**  Configure multiple trustpoints by executing the same command multiple times. A maximum of 10 trustpoints can be configured.

- **Step 8**

  `ocsp-response required`

  **Example:**
  
  Device(ssh-server-cert-profile-user)# ocsp-response required

  (Optional) Mandates the presence of the Online Certificate Status Protocol (OCSP) response with the incoming user certificate.

  **Note**  By default the “no” form of this command is configured and the user certificate is accepted without an OCSP response.
### Purpose

Command or Action | Purpose
---|---
**Step 9**  
end  
**Example:**  
Device(ssh-server-cert-profile-user)# end  
| Exits SSH server certificate profile user configuration mode and enters privileged EXEC mode.

## Verifying Configuration for Server and User Authentication Using Digital Certificates

### SUMMARY STEPS

1. **enable**
2. **show ip ssh**

### DETAILED STEPS

**Step 1**  
**enable**

Enables privileged EXEC mode.

- Enter your password if prompted.

**Example:**  
Device> **enable**

**Step 2**  
**show ip ssh**

Displays the currently configured authentication methods. To confirm the use of certificate-based authentication, ensure that the x509v3-ssh-rsa algorithm is the configured host key algorithm.

**Example:**  
Device# **show ip ssh**

```
SSH Enabled - version 1.99
Authentication methods:publickey,keyboard-interactive,password
Authentication Publickey Algorithms:x509v3-ssh-rsa,ssh-rsa
Hostkey Algorithms:x509v3-ssh-rsa,ssh-rsa
Authentication timeout: 120 secs; Authentication retries: 3
Minimum expected Diffie Hellman key size : 1024 bits
```
Configuration Examples for X.509v3 Certificates for SSH Authentication

Example: Configuring IOS SSH Server to Use Digital Certificates for Server Authentication

Device> enable
Device# configure terminal
Device(config)# ip ssh server algorithm hostkey x509v3-ssh-rsa
Device(config)# ip ssh server certificate profile
Device(ssh-server-cert-profile)# server
Device(ssh-server-cert-profile-server)# trustpoint sign trust1
Device(ssh-server-cert-profile-server)# exit

Example: Configuring IOS SSH Server to Verify User's Digital Certificate for User Authentication

Device> enable
Device# configure terminal
Device(config)# ip ssh server algorithm authentication publickey
Device(config)# ip ssh server algorithm publickey x509v3-ssh-rsa
Device(config)# ip ssh server certificate profile
Device(ssh-server-cert-profile)# user
Device(ssh-server-cert-profile-user)# trustpoint verify trust2
Device(ssh-server-cert-profile-user)# end

Additional References for X.509v3 Certificates for SSH Authentication

<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>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>
Feature Information for X.509v3 Certificates for SSH Authentication

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 Information for X.509v3 Certificates for SSH Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Feature Name</strong></td>
</tr>
<tr>
<td>X.509v3 Certificates for SSH Authentication</td>
</tr>
</tbody>
</table>
SSH Algorithms for Common Criteria Certification

The SSH Algorithms for Common Criteria Certification feature provides the list and order of the algorithms that are allowed for Common Criteria Certification. This module describes how to configure the encryption, Message Authentication Code (MAC), and host key algorithms for a secure shell (SSH) server and client so that SSH connections can be limited on the basis of the allowed algorithms list.

- Finding Feature Information, on page 63
- Information About SSH Algorithms for Common Criteria Certification, on page 63
- How to Configure SSH Algorithms for Common Criteria Certification, on page 65
- Configuration Examples For SSH Algorithms for Common Criteria Certification, on page 70
- Additional References for SSH Algorithms for Common Criteria Certification, on page 71
- Feature Information for SSH Algorithms for Common Criteria Certification, on page 72

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.

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.

Information About SSH Algorithms for Common Criteria Certification

SSH Algorithms for Common Criteria Certification

A Secure Shell (SSH) configuration enables a Cisco IOS SSH server and client to authorize the negotiation of only those algorithms that are configured from the allowed list. If a remote party tries to negotiate using only those algorithms that are not part of the allowed list, the request is rejected and the session is not established.
Cisco IOS SSH Server Algorithms

Cisco IOS secure shell (SSH) servers support the encryption algorithms (Advanced Encryption Standard Counter Mode [AES-CTR], AES Cipher Block Chaining [AES-CBC], Triple Data Encryption Standard [3DES]) in the following order:

1. aes128-ctr
2. aes192-ctr
3. aes256-ctr
4. aes128-cbc
5. 3des-cbc
6. aes192-cbc
7. aes256-cbc

Cisco IOS SSH servers support the Message Authentication Code (MAC) algorithms in the following order:

1. hmac-sha1
2. hmac-sha1-96

Cisco IOS SSH servers support the host key algorithms in the following order:

1. x509v3-ssh-rsa
2. ssh-rsa

Cisco IOS SSH Client Algorithms

Cisco IOS secure shell (SSH) clients support the encryption algorithms (Advanced Encryption Standard counter mode [AES-CTR], AES Cipher Block Chaining [AES-CBC], Triple Data Encryption Standard [3DES]) in the following order:

1. aes128-ctr
2. aes192-ctr
3. aes256-ctr
4. aes128-cbc
5. 3des-cbc
6. aes192-cbc
7. aes256-cbc

Cisco IOS SSH clients support the Message Authentication Code (MAC) algorithms in the following order:

1. hmac-sha1
2. hmac-sha1-96
Cisco IOS SSH clients support only one host key algorithm and do not need a CLI configuration:

- ssh-rsa

# How to Configure SSH Algorithms for Common Criteria Certification

## Configuring an Encryption Key Algorithm for a Cisco IOS SSH Server and Client

### SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `ip ssh {server | client} algorithm encryption {aes128-ctr | aes192-ctr | aes256-ctr | aes128-cbc | 3des-cbc | aes192-cbc | aes256-cbc}`
4. `end`

### 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: <code>Device&gt; enable</code></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: <code>Device# configure terminal</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> ip ssh {server</td>
<td>client} algorithm encryption {aes128-ctr</td>
</tr>
</tbody>
</table>
| Example: `Device(config)# ip ssh server algorithm encryption aes128-ctr aes192-ctr aes256-ctr aes128-cbc 3des-cbc aes192-cbc aes256-cbc`

**Note** The Cisco IOS SSH server and client must have at least one configured encryption algorithm.

**Note** To disable one algorithm from the previously configured algorithm list, use the `no` form of this command. To disable more than one algorithm, use the `no` form of this command multiple times with different algorithm names.
### Troubleshooting Tips

If you try to disable the last encryption algorithm in the configuration, the following message is displayed and the command is rejected:

```
% SSH command rejected: All encryption algorithms cannot be disabled
```

#### Configuring a MAC Algorithm for a Cisco IOS SSH Server and Client

**SUMMARY STEPS**

1. `enable`
2. `configure terminal`
3. `ip ssh {server | client} algorithm mac {hmac-sha1 | hmac-sha1-96}`
4. `end`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><code>enable</code></td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device&gt; enable</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><code>configure terminal</code></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td><code>Device# configure terminal</code></td>
<td></td>
</tr>
</tbody>
</table>
Purpose

Command or Action | Purpose
--- | ---
**Step 3** | ip ssh {server | client} algorithm mac {hmac-sha1 | hmac-sha1-96}
Example: Device(config)# ip ssh server algorithm mac hmac-sha1 hmac-sha1-96
Device(config)# ip ssh client algorithm mac hmac-sha1 hmac-sha1-96

**Step 4** | end
Example: Device(config)# end

Troubleshooting Tips

If you try to disable the last MAC algorithm in the configuration, the following message is displayed and the command is rejected:

```
% SSH command rejected: All mac algorithms cannot be disabled
```

Configuring a Host Key Algorithm for a Cisco IOS SSH Server

SUMMARY STEPS

1. enable
2. configure terminal
3. ip ssh server algorithm hostkey {x509v3-ssh-rsa | ssh-rsa}
4. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>enable</td>
</tr>
</tbody>
</table>
### Troubleshooting Tips

If you try to disable the last host key algorithm in the configuration, the following message is displayed and the command is rejected:

```
% SSH command rejected: All hostkey algorithms cannot be disabled
```
Verifying SSH Algorithms for Common Criteria Certification

SUMMARY STEPS

1. enable
2. show ip ssh

DETAILED STEPS

Step 1  enable
Enable privileged EXEC mode.
• Enter your password if prompted.

Example:
Device> enable

Step 2  show ip ssh
Displays configured Secure Shell (SSH) encryption, host key, and Message Authentication Code (MAC) algorithms.

Example:
The following sample output from the show ip ssh command shows the encryption algorithms configured in the default order:

Device# show ip ssh

The following sample output from the show ip ssh command shows the MAC algorithms configured in the default order:

Device# show ip ssh
MAC Algorithms: hmac-sha1 hmac-sha1-96

The following sample output from the show ip ssh command shows the host key algorithms configured in the default order:

Device# show ip ssh
Hostkey Algorithms: x509v3-ssh-rsa, ssh-rsa
Configuration Examples For SSH Algorithms for Common Criteria Certification

Example: Configuring Encryption Key Algorithms for a Cisco IOS SSH Server

```
Device> enable
Device# configure terminal
Device(config)# ip ssh server algorithm encryption aes128-ctr aes192-ctr aes256-ctr aes128-cbc
   3des-cbc aes192-cbc aes256-cbc
Device(config)# end
```

Example: Configuring Encryption Key Algorithms for a Cisco IOS SSH Client

```
Device> enable
Device# configure terminal
Device(config)# ip ssh client algorithm encryption aes128-ctr aes192-ctr aes256-ctr aes128-cbc
   3des-cbc aes192-cbc aes256-cbc
Device(config)# end
```

Example: Configuring MAC Algorithms for a Cisco IOS SSH Server

```
Device> enable
Device# configure terminal
Device(config)# ip ssh server algorithm mac hmac-sha1 hmac-sha1-96
Device(config)# end
```

Example: Configuring Key Exchange DH Group for a Cisco IOS SSH Server

```
Device> enable
Device# configure terminal
Device(config)# ip ssh server algorithm kex diffie-hellman-group-exchange-shal
Device(config)# end

Device> enable
Device# configure terminal
Device(config)# ip ssh server algorithm kex diffie-hellman-group14-shal
Device(config)# end
```
Example: Configuring Host Key Algorithms for a Cisco IOS SSH Server

Device> enable
Device# configure terminal
Device(config)# ip ssh server algorithm hostkey x509v3-ssh-rsa ssh-rsa
Device(config)# end

Additional References for SSH Algorithms for Common Criteria Certification

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>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>SSH authentication</td>
<td>• Cisco IOS Security Command Reference: Commands S to Z</td>
</tr>
<tr>
<td>X.509v3 digital certificates in server and user authentication</td>
<td>“Secure Shell-Configuring User Authentication Methods” chapter in the Secure Shell Configuration Guide</td>
</tr>
<tr>
<td></td>
<td>“X.509v3 Certificates for SSH Authentication” chapter in the Secure Shell Configuration Guide</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 documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support 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 SSH Algorithms for Common Criteria Certification

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 6: Feature Information for SSH Algorithms for Common Criteria Certification

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSH Algorithms for Common Criteria Certification</td>
<td></td>
<td>The SSH Algorithms for Common Criteria Certification feature provides the list and order of the algorithms that are allowed for Common Criteria Certification. This module describes how to configure the encryption, Message Authentication Code (MAC), and host key algorithms for a secure shell (SSH) server and client so that SSH connections can be limited on the basis of the allowed algorithms list. The following commands were introduced by this feature: `ip ssh {server</td>
</tr>
</tbody>
</table>