Secure Shell Configuration Guide, Cisco IOS Release 15SY

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

CHAPTER 1

Configuring Secure Shell

The Secure Shell (SSH) feature is an application and a protocol that provides a secure replacement to the Berkeley r-tools. The protocol secures sessions using standard cryptographic mechanisms, and the application can be used similarly to the Berkeley rexec and rsh tools. Two versions of SSH are available: SSH Version 1 and SSH Version 2. Unless otherwise noted, the term “SSH” denotes “SSH Version 1” only. For information about SSH Version 2, see the “ Secure Shell Version 2 Support” feature module.

Finding Feature Information

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

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

Prerequisites for Configuring SSH

Note Unless otherwise noted, the term “SSH” denotes “SSH Version 1” only.

- Download the required image on the device. The Secure Shell (SSH) server requires an IPsec (Data Encryption Standard [DES] or 3DES) encryption software image; the SSH client requires an IPsec (DES or 3DES) encryption software image.) For information about downloading a software image, see the *Loading and Managing System Images Configuration Guide*.
- Configure a hostname and host domain for your device by using the **hostname** and **ip domain-name** commands in global configuration mode.
- Generate a Rivest, Shamir, and Adleman (RSA) key pair for your device. This key pair automatically enables SSH and remote authentication when the **crypto key generate rsa** command is entered in global configuration mode.

Note To delete the RSA key pair, use the **crypto key zeroize rsa** global configuration command. Once you delete the RSA key pair, you automatically disable the SSH server.

- Configure user authentication for local or remote access. You can configure authentication with or without authentication, authorization, and accounting (AAA). For more information, see the *Authentication, Authorization, and Accounting Configuration Guide*.

Restrictions for Configuring SSH

Note Unless otherwise noted, the term “SSH” denotes “SSH Version 1” only.

- The Secure Shell (SSH) server and SSH client are supported on Data Encryption Standard (DES) (56-bit) and 3DES (168-bit) data encryption software images only. In DES software images, DES is the only encryption algorithm available. In 3DES software images, both DES and 3DES encryption algorithms are available.
- Execution shell is the only application supported.
- The login banner is not supported in Secure Shell Version 1. It is supported in Secure Shell Version 2.

Information About Secure Shell (SSH)

Note Unless otherwise noted, the term “SSH” denotes “SSH Version 1” only.

SSH Server

Note Unless otherwise noted, the term “SSH” denotes “SSH Version 1” only.

The Secure Shell (SSH) Server feature enables an SSH client to make a secure, encrypted connection to a Cisco device. This connection provides functionality that is similar to that of an inbound Telnet connection. Before SSH, security was limited to Telnet security. SSH allows a strong encryption to be used with the Cisco software authentication. The SSH server in Cisco software works with publicly and commercially available SSH clients.

SSH Integrated Client

Note Unless otherwise noted, the term “SSH” denotes “SSH Version 1” only.

The Secure Shell (SSH) Integrated Client feature is an application that runs over the SSH protocol to provide device authentication and encryption. The SSH client enables a Cisco device to make a secure, encrypted connection to another Cisco device or to any other device running the SSH server. This connection provides functionality similar to that of an outbound Telnet connection except that the connection is encrypted. With authentication and encryption, the SSH client allows for secure communication over an unsecured network.

The SSH client in Cisco software works with publicly and commercially available SSH servers. The SSH client supports the ciphers of Data Encryption Standard (DES), 3DES, and password authentication. User authentication is performed like that in the Telnet session to the device. The user authentication mechanisms supported for SSH are RADIUS, TACACS+, and the use of locally stored usernames and passwords.

Note The SSH client functionality is available only when the SSH server is enabled.

RSA Authentication Support

Rivest, Shamir, and Adleman (RSA) authentication available in Secure Shell (SSH) clients is not supported on the SSH server for Cisco software by default. For more information about RSA authentication support, see the “Configuring a Router for SSH Version 2 Using RSA Pairs” section of the “Secure Shell Version 2 Support” module.

How to Configure SSH

Configuring an SSH Server

Note Unless otherwise noted, the term “SSH” denotes “SSH Version 1” only.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip ssh** {**timeout** *seconds* | **authentication-retries** *integer*}
4. **ip ssh rekey** {**time** *time* | **volume** *volume*}
5. **exit**
6. **show ip ssh**

DETAILED STEPS

	Command or Action	Purpose
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** {**timeout** *seconds* | **authentication-retries** *integer*}
	
	Example:
	`Device(config)# ip ssh timeout 30`
		Configures Secure Shell (SSH) control parameters.
		Note This command can also be used to establish the number of password prompts provided to the user. The number is the lower of the following two values:
		• Value proposed by the client using the **ssh -o numberofpasswordprompt** command.
		• Value configured on the device using the **ip ssh authentication-retries** *integer*command, plus one.

	Command or Action	Purpose

Step 4	**ip ssh rekey** {**time** *time* | **volume** *volume*}
	
	Example:
	`Device(config)# ip ssh rekey time 108`
		(Optional) Configures a time-based rekey or a volume-based rekey for SSH.

Step 5	**exit** **Example:** `Device(config)# exit`	Returns to privileged EXEC mode.
Step 6	**show ip ssh** **Example:** `Device# show ip ssh`	(Optional) Verifies that the SSH server is enabled and displays the version and configuration data for the SSH connection.

Invoking an SSH Client

Note Unless otherwise noted, the term "SSH" denotes "SSH Version 1" only.

Perform this task to invoke the Secure Shell (SSH) client. The SSH client runs in user EXEC mode and has no specific configuration tasks.

SUMMARY STEPS

1. **enable**
2. **ssh -l** *username* **-vrf** *vrf-name* *ip-address*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**ssh -l** *username* **-vrf** *vrf-name* *ip-address* **Example:** `Device# ssh -l user1 -vrf vrf1 192.0.2.1`	Invokes the SSH client to connect to an IP host or address in the specified virtual routing and forwarding (VRF) instance.

Troubleshooting Tips

Note Unless otherwise noted, the term "SSH" denotes "SSH Version 1" only.

- If your Secure Shell (SSH) configuration commands are rejected as illegal commands, you have not successfully generated an Rivest, Shamir, and Adleman (RSA) key pair for your device. Make sure that you have specified a hostname and domain. Then use the **crypto key generate rsa** command to generate an RSA key pair and enable the SSH server.
- When configuring the RSA key pair, you might encounter the following error messages:
  - No hostname specified.
    You must configure a hostname for the device using the **hostname** global configuration command. See the "IPsec and Quality of Service" module for more information.
  - No domain specified.
    You must configure a host domain for the device using the **ip domain-name** global configuration command. See the "IPsec and Quality of Service" module for more information
- The number of allowable SSH connections is limited to the maximum number of vtys configured for the device. Each SSH connection uses a vty resource.
- SSH uses either local security or the security protocol that is configured through AAA on your device for user authentication. When configuring Authentication, Authorization, and Accounting (AAA), you must ensure that AAA is disabled on the console for user authentication. AAA authorization is disabled on the console by default. If AAA authorization is enabled on the console, disable it by configuring the **no aaa authorization console** command during the AAA configuration stage.

Configuration Examples for SSH

Example: Configuring an SSH Server

Note Unless otherwise noted, the term "SSH" denotes "SSH Version 1" only.

The following is an example of the Secure Shell (SSH) control parameters configured for the server. In this example, the timeout interval of 30 seconds has been specified. This timeout interval is used during the SSH negotiation phase.

```
Device> enable
Device# configure terminal
```

```
Device(config)# ip ssh timeout 30
Device(config)# end
```

Example: Invoking an SSH Client

Note Unless otherwise noted, the term “SSH” denotes “SSH Version 1” only.

In the following example, the Secure Shell (SSH) client has been invoked to connect to IP address 192.0.2.1 in the specified virtual routing and forwarding (VRF) instance:

```
Device> enable
Device# configure terminal
Device(config)# ssh -1 user1 -vrf vrf1 192.0.2.1
Device(config)# end
```

Example: Verifying SSH

Note Unless otherwise noted, the term “SSH” denotes “SSH Version 1” only.

To verify that the Secure Shell (SSH) server is enabled and to display the version and configuration data for your SSH connection, use the **show ip ssh** command. The following example shows that SSH is enabled:

```
Device# show ip ssh

SSH Enabled - version 1.5
Authentication timeout: 120 secs; Authentication retries: 3
```

The following example shows that SSH is disabled:

```
Device# show ip ssh

%SSH has not been enabled
```

To verify the status of your SSH server connections, use the **show ssh** command. The following example shows the SSH server connections on the device when SSH is enabled:

```
Device# show ssh

Connection      Version     Encryption State Username
 0 1.5 3DES Session Started  guest
```

The following example shows that SSH is disabled:

```
Device# show ssh

%No SSH server connections running.
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
Authentication, authorization, and accounting (AAA)	*Authentication, Authorization, and Accounting Configuration Guide*
IPsec	"IPsec and Quality of Service" module
SSH Version 2	"Secure Shell Version 2 Support" module
Downloading a software image	*Loading and Managing System Images Configuration Guide*

Technical Assistance

Description	Link
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.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Configuring Secure Shell

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 Configuring Secure Shell

Feature Name	Releases	Feature Information
Secure Shell	12.0(5)S 15.0(2)SE 15.1(1)SY	The Secure Shell (SSH) feature is an application and a protocol that provides a secure replacement to the Berkeley r-tools. The protocol secures sessions using standard cryptographic mechanisms, and the application can be used similarly to the Berkeley rexec and rsh tools. Two versions of SSH are available: SSH Version 1 and SSH Version 2. This document describes SSH Version 1. This document also includes information about the Secure Shell SSH Version 1 Integrated Client feature and the Secure Shell SSH Version 1 Server Support feature. Both features are part of the Secure Shell functionality.

CHAPTER 2

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

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

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to 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 **-l** 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 12.

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**
9. **ssh -l** *userid* **:** {*number*} {*ip-address*}

DETAILED STEPS

	Command or Action	Purpose
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	**line** *line-number* *ending-line-number* **Example:** `Device# line 1 3`	Identifies a line for configuration and enters line configuration mode.
Step 4	**no exec** **Example:** `Device(config-line)# no exec`	Disables EXEC processing on a line.
Step 5	**login authentication** *listname* **Example:** `Device(config-line)# login authentication default`	Defines a login authentication mechanism for the lines. **Note** The authentication method must use a username and password.

	Command or Action	Purpose
Step 6	**transport input ssh** **Example:** `Device(config-line)# transport input ssh`	Defines which protocols to use to connect to a specific line of the device. • The **ssh** keyword must be used for the Reverse SSH Enhancements feature.
Step 7	**exit** **Example:** `Device(config-line)# exit`	Exits line configuration mode.
Step 8	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode.

Step 9	**ssh -l** *userid* **:** {*number*} {*ip-address*}
	
	Example:
	
	`Device# ssh -l lab:1 router.example.com`
		Specifies the user ID to use when logging in on the remote networking device that is running the SSH server.
		
		• *userid* --User ID.
		• **:** --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

	Command or Action	Purpose
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	**line** *line-number* *ending-line-number* **Example:** `Device# line 1 200`	Identifies a line for configuration and enters line configuration mode.
Step 4	**no exec** **Example:** `Device(config-line)# no exec`	Disables EXEC processing on a line.
Step 5	**login authentication** *listname* **Example:** `Device(config-line)# login authentication default`	Defines a login authentication mechanism for the lines. **Note** The authentication method must use a username and password.

	Command or Action	Purpose
Step 6	**rotary** *group* **Example:** `Device(config-line)# rotary 1`	Defines a group of lines consisting of one or more virtual terminal lines or one auxiliary port line.
Step 7	**transport input ssh** **Example:** `Device(config-line)# transport input ssh`	Defines which protocols to use to connect to a specific line of the device. • The **ssh** keyword must be used for the Reverse SSH Enhancements feature.
Step 8	**exit** **Example:** `Device(config-line)# exit`	Exits line configuration mode.
Step 9	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode.

Step 10	**ssh -l** *userid* **:rotary** {*number*} {*ip-address*}
	
	Example:
	
	`Device# ssh -l lab:rotary1 router.example.com`
		Specifies the user ID to use when logging in on the remote networking device that is running the SSH server.
		• *userid* --User ID.
		• **:** --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.

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

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**debug ip ssh client** **Example:** `Device# debug ip ssh client`	Displays debugging messages for the SSH client.

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

	Command or Action	Purpose
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`	Displays debugging messages for the SSH server.

	Command or Action	Purpose
Step 3	**show ssh** **Example:** `Device# show ssh`	Displays the status of the SSH server connections.
Step 4	**show line** **Example:** `Device# show line`	Displays parameters of a terminal line.

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:

```
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:

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

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Configuring Secure Shell	Secure Shell Configuration Guide
Security commands	Cisco IOS Security Command Reference

Technical Assistance

Description	Link
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.	http://www.cisco.com/cisco/web/support/index.html

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Configuring Secure Shell	Secure Shell Configuration Guide
Security commands	Cisco IOS Security Command Reference

Standards

Standards	Title
No new or modified standards are supported by this feature.	--

MIBs

MIBs	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
None	--

Technical Assistance

Description	Link
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.	http://www.cisco.com/cisco/web/support/index.html

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 2: Feature Information for Reverse SSH Enhancements

Feature Name	Releases	Feature Information
Reverse SSH Enhancements	Cisco IOS 12.3(11)T	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: **ssh**.

CHAPTER 3

Secure Copy

The Secure Copy (SCP) feature provides a secure and authenticated method for copying device configurations or device image files. SCP relies on Secure Shell (SSH), an application and protocol that provide a secure replacement for the Berkeley r-tools suite (Berkeley university's own set of networking applications). This document provides the procedure to configure a Cisco device for SCP server-side functionality.

Finding Feature Information

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

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

Prerequisites for Secure Copy

- Before enabling Secure Copy (SCP), you must correctly configure Secure Shell (SSH), authentication, and authorization on the device.

- Because SCP relies on SSH for its secure transport, the device must have a Rivest, Shamir, and Adelman (RSA) key pair.

Information About Secure Copy

How Secure Copy Works

The behavior of Secure Copy (SCP) is similar to that of remote copy (RCP), which comes from the Berkeley r-tools suite (Berkeley university's own set of networking applications), except that SCP relies on Secure Shell (SSH) for security. In addition, SCP requires that authentication, authorization, and accounting (AAA) authorization be configured so that the device can determine whether the user has the correct privilege level.

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

Note Enable the SCP option while using the pscp.exe file with the Cisco software.

How to Configure Secure Copy

Configuring Secure Copy

To configure a Cisco device for Secure Copy (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**
8. **exit**
9. **show running-config**
10. **debug ip scp**

DETAILED STEPS

	Command or Action	Purpose
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`	Sets AAA authentication at login.
Step 4	**aaa authentication login** {**default** | *list-name*} *method1* [*method2...*] **Example:** `Device(config)# aaa authentication login default group tacacs+`	Enables the AAA access control system.
Step 5	**aaa authorization** {**network** | **exec** | **commands** *level* | **reverse-access** | **configuration**} {**default** | *list-name*} [*method1* [*method2...*]] **Example:** `Device(config)# aaa authorization exec default group tacacs+`	Sets parameters that restrict user access to a network. **Note** The **exec** keyword runs authorization to determine if the user is allowed to run an EXEC shell; therefore, you must use the **exec** keyword when you configure SCP.
Step 6	**username** *name* [**privilege** *level*] **password** *encryption-type encrypted-password* **Example:** `Device(config)# username superuser privilege 2 password 0 superpassword`	Establishes a username-based authentication system. **Note** You may omit this step if a network-based authentication mechanism, such as TACACS+ or RADIUS, has been configured.
Step 7	**ip scp server enable** **Example:** `Device(config)# ip scp server enable`	Enables SCP server-side functionality.

	Command or Action	Purpose
Step 8	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.
Step 9	**show running-config** **Example:** `Device# show running-config`	(Optional) Displays the SCP server-side functionality.
Step 10	**debug ip scp** **Example:** `Device# debug ip scp`	(Optional) Troubleshoots SCP authentication problems.

Configuration Examples for Secure Copy

Example: Secure Copy Configuration Using Local Authentication

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

```
! AAA authentication and authorization must be configured properly in order for SCP to work.
aaa new-model
aaa authentication login default local
aaa authorization exec default local
username user1 privilege 15 password 0 lab
! SSH must be configured and functioning properly.
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

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
Secure Shell Version 1 and 2 support	*Secure Shell Configuration Guide*
Authentication and authorization commands	Cisco IOS Security Command Reference: Commands A to C
Configuring authentication and authorization	*Authentication, Authorization, and Accounting Configuration Guide*

Technical Assistance

Description	Link
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.	http://www.cisco.com/cisco/web/support/index.html

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 3: Feature Information for Secure Copy

Feature Name	Releases	Feature Information
Secure Copy	Cisco IOS 12.0(21)S Cisco IOS 12.2(2)T Cisco IOS 12.2(25)S	The Secure Copy (SCP) feature provides a secure and authenticated method for copying device configurations or device image files. SCP relies on Secure Shell (SSH), an application and protocol that provide a secure replacement for the Berkeley r-tools suite. The following commands were introduced or modified: **debug ip scp**, **ip scp server enable**.

Glossary

AAA—authentication, authorization, and accounting. A 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. Relies on Remote Shell (Berkeley r-tools suite) for security; RCP copies files such as device images and startup configurations to and from devices.

SCP—secure copy. Relies on SSH for security; SCP support allows secure and authenticated copying of anything that exists in the Cisco IOS File System (IFS). SCP is derived from RCP.

SSH—Secure Shell. An application and protocol that provide a secure replacement for the Berkeley r-tools suite. The protocol secures the sessions using standard cryptographic mechanisms, and the application can be used similar to the Berkeley rexec and rsh tools. SSH Version 1 is implemented in the Cisco software.

CHAPTER 4

VRF-Aware SCP

The VRF-Aware SCP feature applies the secure copy protocol (SCP) functionality to Virtual Routing and Forwarding (VRF) interfaces using the Secure Shell (SSH) application to copy device configurations or device image files.

Finding Feature Information

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

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

Prerequisites for VRF-Aware SCP

- Ensure that Secure Shell (SSH) connection is enabled.
- Ensure that Virtual Routing and Forwarding (VRF) configuration is available on the device.

Information About VRF-Aware SCP

SCP and SSH

The secure copy protocol (SCP) feature allows a user with appropriate authorization to copy any file that exists in the Cisco IOS File System (IFS) to and from a device by using the copy command. Being Virtual Routing and Forwarding (VRF) aware, the SCP feature can provide the service only to a specific group or interface rather than providing global access and configuration. The VRF-aware SCP feature enables administrators to have more control and added security.

SCP relies on Secure Shell (SSH) for security and authentication.

Use the **ip ssh source-interface** command to source SSH traffic from any interface, including a VRF interface.

How to Configure VRF-Aware SCP

Configuring SCP to Use VRF-Aware Interface

Before You Begin

Configure Virtual Routing and Forwarding (VRF) aware interfaces.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip ssh source-interface** *interface*
4. **exit**
5. **copy running-config scp://***username@destination-host-address*[*/destination-directory*][*/destination-filename*]
6. **copy scp://***username@source-host-address*[*/source-directory*][*/source-filename*] **bootflash:**
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip ssh source-interface** *interface* **Example:** `Device(config)# ip ssh source-interface GigabitEthernet 1/1`	Specifies the IP address of an interface as the source address for a Secure Shell (SSH) client device. Provide a VRF-aware interface to use the VRF-Aware SCP feature.
Step 4	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.
Step 5	**copy running-config scp:**//*username*@*destination-host-address*[/*destination-directory*][/*destination-filename*] **Example:** `Device# copy running-config scp://guest@10.76.76.160/router.cfg`	Copies a file from the current running configuration file of the source device to a destination device with secure copy protocol (SCP).
Step 6	**copy scp:**//*username*@*source-host-address*[/*source-directory*][/*source-filename*] **bootflash:** **Example:** `Device# copy scp://guest@10.76.76.160/router.cfg bootflash:`	Copies a file to the boot flash memory of the destination device from the source device with SCP.
Step 7	**exit** **Example:** `Device# exit`	Exits privileged EXEC mode and returns to user EXEC mode.

Configuration Examples for VRF-Aware SCP

Example: Configuring SCP Using VRF-Aware Interface

```
Device> enable
Device# configure terminal
Device(config)# ip ssh source-interface GigabitEthernet 1/1
Device(config)# exit
Device# copy running-config scp://guest@10.76.76.160/router.cfg

Address or name of remote host [10.76.76.160]?
Destination username [guest]?
Destination filename [router.cfg]?
Writing router.cfg Password:
!
Sink: C0644 2574 router.cfg
2574 bytes copied in 20.852 secs (123 bytes/sec)

Device# copy scp://guest@10.76.76.160/router.cfg bootflash:

Destination filename [router.cfg]?
Password:
Sending file modes: C0644 2574 router.cfg
!
2574 bytes copied in 17.975 secs (143 bytes/sec)

Device# exit
```

Additional References for VRF-Aware SCP

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
Security commands	• Cisco IOS Security Command Reference: Commands A to C • Cisco IOS Security Command Reference: Commands D to L • Cisco IOS Security Command Reference: Commands M to R • Cisco IOS Security Command Reference: Commands S to Z
Secure Copy Protocol	"Secure Copy" module in the *Secure Shell Configuration Guide* publication

Technical Assistance

Description	Link
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.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for VRF-Aware SCP

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 VRF-Aware SCP

Feature Name	Releases	Feature Information
VRF-Aware SCP	15.2(1)SY	The VRF-Aware SCP feature applies the secure copy protocol (SCP) functionality to Virtual Routing and Forwarding (VRF) interfaces using the Secure Shell (SSH) application to copy device configurations or device image files. The following commands were introduced or modified by this feature: **ip ssh source-interface**.

CHAPTER 5

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

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

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to 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.

Note The SSH Version 2 server and the SSH Version 2 client are supported on your Cisco software, depending on your release. (The SSH client runs both the SSH Version 1 protocol and the SSH Version 2 protocol. The SSH client is supported in 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 Secure Shell Version 2 Support feature allows you to configure SSH 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. For an example of an SNMP trap generation configuration, see the "" section.

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. For an example of SNMP debugging, see the " Example: SNMP Debugging section.

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 56" 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

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**hostname** *name* **Example:** `Device(config)# hostname cisco7200`	Configures a hostname for your device.
Step 4	**ip domain-name** *name* **Example:** `cisco7200(config)# ip domain-name example.com`	Configures a domain name for your device.
Step 5	**crypto key generate rsa** **Example:** `cisco7200(config)# crypto key generate rsa`	Enables the SSH server for local and remote authentication.
Step 6	**ip ssh** [**time-out** *seconds* | **authentication-retries** *integer*] **Example:** `cisco7200(config)# ip ssh time-out 120`	(Optional) Configures SSH control variables on your device.
Step 7	**ip ssh version** [**1** | **2**] **Example:** `cisco7200(config)# ip ssh version 1`	(Optional) Specifies the version of SSH to be run on your device.
Step 8	**exit** **Example:** `cisco7200(config)# exit`	Exits global configuration mode and enters privileged EXEC mode. • Use **no hostname** command to return to the default host.

Configuring a Device for SSH Version 2 Using RSA Key Pairs

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
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.
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*]	Configures SSH control variables on your device.

	Command or Action	Purpose
	Example: `Device(config)# ip ssh time-out 12`	
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

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
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.

	Command or Action	Purpose
		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

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
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*	(Optional) Specifies the SSH key type and version.

	Command or Action	Purpose
	Example: `host1(conf-ssh-pubkey-server)# key-hash ssh-rsa key1`	• 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:
	`host1(conf-ssh-pubkey-server)# end`
		Exits public-key server configuration mode and returns to privileged EXEC mode.

Step 12	**configure terminal** **Example:** `host1# configure terminal`	Enters global configuration mode.
Step 13	**ip ssh stricthostkeycheck** **Example:** `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

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

SUMMARY STEPS

1. **ssh** [**-v** {**1** | **2**} | **-c** {**aes128-ctr** | **aes192-ctr** | **aes256-ctr** | **aes128-cbc** | **3des** | **aes192-cbc** | **aes256-cbc**} | **-l** *user-id* | **-l** *user-id***:***vrf-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**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ssh** [**-v** {**1** | **2**} | **-c** {**aes128-ctr** | **aes192-ctr** | **aes256-ctr** | **aes128-cbc** | **3des** | **aes192-cbc** | **aes256-cbc**} | **-l** *user-id* | **-l** *user-id***:***vrf-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:** `Device# ssh -v 2 -c aes256-ctr -m hmac-sha1-96 -l user2 10.76.82.24`	Starts an encrypted session with a remote networking device.

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.

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
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.

	Command or Action	Purpose
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 scpserverenable** **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

SUMMARY STEPS

1. **enable**
2. **show ssh**
3. **exit**

DETAILED STEPS

	Command or Action	Purpose
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
1          2.0     IN   aes128-cbc  hmac-md5     Session started       lab
1          2.0     OUT  aes128-cbc  hmac-md5     Session started       lab
-------------------------------------------------------------------------
```

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

```
-------------------------------------------------------------------------
Device# show ssh

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

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

```
-------------------------------------------------------------------------
Device# show ssh
```

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

Verifying the Secure Shell Status

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
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

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
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: kex: client->server aes128-cbc hmac-md5 none
00:33:55: SSH2: kex: server->client aes128-cbc hmac-md5 none
00:33:55: SSH2 1: expecting SSH2_MSG_KEXDH_INIT
00:33:55: SSH2 1: ssh_receive: 144 bytes received
00:33:55: SSH2 1: input: packet len 144
00:33:55: SSH2 1: partial packet 8, need 136, maclen 0
00:33:55: SSH2 1: input: padlen 5
00:33:55: SSH2 1: received packet type 30
00:33:55: SSH2 1: SSH2_MSG_KEXDH_INIT received
00:33:55: SSH2 1: signature length 111
00:33:55: SSH2 1: send: len 384 (includes padlen 7)
00:33:55: SSH2: 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: send: len 32 (includes padlen 10)
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: send: len 32 (includes padlen 13)
00:33:56: SSH2 1: done calc MAC out #4
00:34:04: SSH2 1: ssh_receive: 160 bytes received
00:34:04: SSH2 1: input: packet len 64
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #5 ok
00:34:04: SSH2 1: input: padlen 13
00:34:04: SSH2 1: received packet type 50
00:34:04: SSH2 1: send: len 16 (includes padlen 10)
00:34:04: SSH2 1: done calc MAC out #5
00:34:04: SSH2 1: authentication successful for lab
00:34:04: SSH2 1: input: packet len 64
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #6 ok
00:34:04: SSH2 1: input: padlen 6
```

```
00:34:04: SSH2 1: received packet type 2
00:34:04: SSH2 1: ssh_receive: 64 bytes received
00:34:04: SSH2 1: input: packet len 48
00:34:04: SSH2 1: partial packet 16, need 32, maclen 16
00:34:04: SSH2 1: MAC #7 ok
00:34:04: SSH2 1: input: padlen 19
00:34:04: SSH2 1: received packet type 90
00:34:04: SSH2 1: channel open request
00:34:04: SSH2 1: send: len 32 (includes padlen 10)
00:34:04: SSH2 1: done calc MAC out #6
00:34:04: SSH2 1: ssh_receive: 192 bytes received
00:34:04: SSH2 1: input: packet len 64
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #8 ok
00:34:04: SSH2 1: input: padlen 13
00:34:04: SSH2 1: received packet type 98
00:34:04: SSH2 1: pty-req 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-req 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
00:34:07: SSH2 1: MAC #13 ok
00:34:07: SSH2 1: input: padlen 17
00:34:07: SSH2 1: received packet type 94
00:34:07: SSH2 1: send: len 32 (includes padlen 17)
00:34:07: SSH2 1: done calc MAC out #10
00:34:08: SSH2 1: ssh_receive: 48 bytes received
00:34:08: SSH2 1: input: packet len 32
00:34:08: SSH2 1: partial packet 16, need 16, maclen 16
00:34:08: SSH2 1: MAC #14 ok
00:34:08: SSH2 1: input: padlen 17
00:34:08: SSH2 1: received packet type 94
00:34:08: SSH2 1: send: len 32 (includes padlen 17)
00:34:08: SSH2 1: done calc MAC out #11
00:34:08: SSH2 1: ssh_receive: 48 bytes received
00:34:08: SSH2 1: input: packet len 32
00:34:08: SSH2 1: partial packet 16, need 16, maclen 16
00:34:08: SSH2 1: MAC #15 ok
00:34:08: SSH2 1: input: padlen 17
```

```
00:34:08: SSH2 1: received packet type 94
00:34:08: SSH2 1: send: len 32 (includes padlen 16)
00:34:08: SSH2 1: done calc MAC out #12
00:34:08: SSH2 1: send: len 48 (includes padlen 18)
00:34:08: SSH2 1: done calc MAC out #13
00:34:08: SSH2 1: send: len 16 (includes padlen 6)
00:34:08: SSH2 1: done calc MAC out #14
00:34:08: SSH2 1: send: len 16 (includes padlen 6)
00:34:08: SSH2 1: done calc MAC out #15
00:34:08: SSH1: Session terminated normally
```

Configuration Examples for Secure Shell Version 2 Support

Example: Configuring Secure Shell Version 1

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

Example: Configuring Secure Shell Version 2

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

Example: Configuring Secure Shell Versions 1 and 2

```
Router# configure terminal
Router(config)# no ip ssh version
```

Example: Starting an Encrypted Session with a Remote Device

```
Device# ssh -v 2 -c aes256-cbc -m hmac-sha1-160 -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 58" section.

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

Examples: SSH Keyboard Interactive Authentication

Example: Enabling Client-Side Debugs

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

```
Password:
Password:
Password:
Password:
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

Password:
*Nov 17 12:50:53.199: SSH0: sent protocol version id SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.199: SSH CLIENT0: protocol version id is - SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.199: SSH CLIENT0: sent protocol version id SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.199: SSH CLIENT0: protocol version exchange successful
*Nov 17 12:50:53.203: SSH0: protocol version id is - SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.335: SSH CLIENT0: key exchange successful and encryption on
*Nov 17 12:50:53.335: SSH2 CLIENT 0: using method keyboard-interactive
Password:
Password:
Password:
*Nov 17 12:51:01.887: SSH2 CLIENT 0: using method password authentication
Password:
Password: lab
Device2>

*Nov 17 12:51:11.407: SSH2 CLIENT 0: SSH2_MSG_USERAUTH_SUCCESS message received
*Nov 17 12:51:11.407: SSH CLIENT0: user authenticated
*Nov 17 12:51:11.407: SSH2 CLIENT 0: pty-req request sent
*Nov 17 12:51:11.411: SSH2 CLIENT 0: shell request sent
*Nov 17 12:51:11.411: SSH CLIENT0: session open
```

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:cisco1
Your password has expired.
Enter a new one now.
New Password: cisco
Re-enter New password: cisco12
The New and Re-entered passwords have to be the same.
Try again.
New Password: cisco
Re-enter New password: cisco

Device2>
```

Example: Enabling ChPass and Expiring the Password After Three Logins

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

```
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

Device2>
```

Example: SNMP Debugging

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

```
Device1# debug snmp packet

SNMP packet debugging is on
Device1# ssh -l lab 10.0.0.2
Password:

Device2# exit

[Connection to 10.0.0.2 closed by foreign host]
Device1#
*Jul 18 10:18:42.619: SNMP: Queuing packet to 10.0.0.2
*Jul 18 10:18:42.619: SNMP: V1 Trap, ent cisco, addr 10.0.0.1, gentrap 6, spectrap 1
local.9.3.1.1.2.1 = 6
tcpConnEntry.1.10.0.0.1.22.10.0.0.2.55246 = 4
ltcpConnEntry.5.10.0.0.1.22.10.0.0.2.55246 = 1015
ltcpConnEntry.1.10.0.0.1.22.10.0.0.2.55246 = 1056
ltcpConnEntry.2.10.0.0.1.22.10.0.0.2.55246 = 1392
local.9.2.1.18.2 = lab
*Jul 18 10:18:42.879: SNMP: Packet sent via UDP to 10.0.0.2

Device1#
```

Examples: SSH Debugging Enhancements

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

```
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:kex: client->server enc:aes128-cbc mac:hmac-sha1
```

```
00:04:22: SSH2:kex: server->client enc:aes128-cbc mac:hmac-sha1
00:04:22: SSH2 0: expecting SSH2_MSG_KEXDH_INIT
00:04:22: SSH2 0: SSH2_MSG_KEXDH_INIT received
00:04:22: SSH2: kex_derive_keys complete
00:04:22: SSH2 0: SSH2_MSG_NEWKEYS sent
00:04:22: SSH2 0: waiting for SSH2_MSG_NEWKEYS
00:04:22: SSH2 0: SSH2_MSG_NEWKEYS received
00:04:24: SSH2 0: authentication successful for lab
00:04:24: SSH2 0: channel open request
00:04:24: SSH2 0: pty-req request
00:04:24: SSH2 0: setting TTY - requested: height 24, width 80; set: height 24, width 80
00:04:24: SSH2 0: shell request
00:04:24: SSH2 0: shell message received
00:04:24: SSH2 0: starting shell for vty
00:04:38: SSH0: Session terminated normally
```

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

```
Device# debug ip ssh packet

00:05:43: SSH2 0: send:packet of  length 280 (length also includes padlen of 4)
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: input: total packet length of 280 bytes
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 24 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: input: padlength 4 bytes
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: input: total packet length of 144 bytes
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 16 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0
00:05:43: SSH2 0: input: padlength 6 bytes
00:05:43: SSH2 0: signature length 143
00:05:43: SSH2 0: send:packet of  length 448 (length also includes padlen of 7)
00:05:43: SSH2 0: send:packet of  length 16 (length also includes padlen of 10)
00:05:43: SSH2 0: newkeys: mode 1
00:05:43: SSH2 0: ssh_receive: 16 bytes received
00:05:43: SSH2 0: input: total packet length of 16 bytes
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 8 bytes, maclen 0
00:05:43: SSH2 0: input: padlength 10 bytes
00:05:43: SSH2 0: newkeys: mode 0
00:05:43: SSH2 0: ssh_receive: 52 bytes received
00:05:43: SSH2 0: input: total packet length of 32 bytes
00:05:43: SSH2 0: partial packet length(block size)16 bytes,needed 16 bytes, maclen 20
00:05:43: SSH2 0: MAC compared for #3 :ok
```

Additional References for Secure Shell Version 2 Support

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases

Related Topic	Document Title
AAA Hostname and host domain configuration tasks Secure shell configuration tasks	*Security Configuration Guide: Securing User Services*
Downloading a software image Configuration fundamentals	*Configuration Fundamentals Configuration Guide*
IPsec configuration tasks	*Security Configuration Guide: Secure Connectivity*
SNMP traps configuration tasks	*SNMP Configuration Guide*

Standards

Standards	Title
IETF Secure Shell Version 2 Draft Standards	Internet Engineering Task Force website

Technical Assistance

Description	Link
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.	http://www.cisco.com/cisco/web/support/index.html

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 5: Feature Information for Secure Shell Version 2 Support

Feature Name	Releases	Feature Information
Secure Shell Version 2 Support	Cisco IOS 12.2(11)T Cisco IOS 12.2(25)S Cisco IOS 12.3(4)T Cisco IOS 15.3(2)S	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**.
Secure Shell Version 2 Client and Server Support	Cisco IOS 12.0(32)SY Cisco IOS 12.3(7)JA Cisco IOS 12.4(17)	The Cisco IOS image was updated to provide for the automatic generation of SNMP traps when an SSH session terminates.
SSH Keyboard Interactive Authentication	Cisco IOS 12.2(33)SXH3 Cisco IOS 12.4(18)	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.

Feature Name	Releases	Feature Information
Secure Shell Version 2 Enhancements	Cisco IOS 12.2(50)SY Cisco IOS 12.4(20)T Cisco IOS 15.1(2)S	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. In Cisco IOS 15.1(2)S, support was added for the Cisco 7600 series router. **Note** Only the VRF-aware SSH feature is supported in Cisco IOS Release 12.2(50)SY. The following commands were introduced or modified: **debug ip ssh**, **ip ssh dh min size**.
Secure Shell Version 2 Enhancements for RSA Keys.	Cisco IOS 15.0(1)M Cisco IOS 15.1(1)S	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: **ip ssh pubkey-chain**, **ip ssh stricthostkeycheck**.

CHAPTER 6

SSH Terminal-Line Access

The SSH Terminal-Line Access feature provides users secure access to tty (text telephone) lines. tty allows the hearing- and speech-impaired to communicate by using a telephone to type messages.

Finding Feature Information

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

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

Prerequisites for SSH Terminal-Line Access

Download the required image to your router. The secure shell (SSH) server requires the router to have an IPSec (Data Encryption Standard (DES) or 3DES) encryption software image from Cisco IOS Release 12.1(1)T or a later release. The SSH client requires the router to have an IPSec (DES or 3DES) encryption software image from Cisco IOS Release 12.1(3)T or a later release. See the *Cisco IOS Configuration Fundamentals Configuration Guide* , Release 12.4T for more information on downloading a software image.

The SSH server requires the use of a username and password, which must be defined through the use of a local username and password, TACACS+, or RADIUS.

Note The SSH Terminal-Line Access feature is available on any image that contains SSH.

Restrictions for SSH Terminal-Line Access

Console Server Requirement

To configure secure console server access, you must define each line in its own rotary and configure SSH to use SSH over the network when user want to access each of those devices.

Memory and Performance Impact

Replacing reverse Telnet with SSH may reduce the performance of available tty lines due to the addition of encryption and decryption processing above the vty processing. (Any cryptographic mechanism uses more memory than a regular access.)

Information About SSH Terminal-Line Access

Overview of SSH Terminal-Line Access

Cisco IOS supports reverse Telnet, which allows users to Telnet through the router--via a certain port range--to connect them to tty (asynchronous) lines. Reverse Telnet has allowed users to connect to the console ports of remote devices that do not natively support Telnet. However, this method has provided very little security because all Telnet traffic goes over the network in the clear. The SSH Terminal-Line Access feature replaces reverse Telnet with SSH. This feature may be configured to use encryption to access devices on the tty lines, which provide users with connections that support strong privacy and session integrity.

SSH is an application and a protocol that provides secure replacement for the suite of Berkeley r-tools such as rsh, rlogin, and rcp. (Cisco IOS supports rlogin.) The protocol secures the sessions using standard cryptographic mechanisms, and the application can be used similarly to the Berkeley rexec and rsh tools. Currently two versions of SSH are available: SSH Version 1 and SSH Version 2. Only SSH Version 1 is implemented in the Cisco IOS software.

The SSH Terminal-Line Access feature enables users to configure their router with secure access and perform the following tasks:

- Connect to a router that has multiple terminal lines connected to consoles or serial ports of other routers, switches, or devices.
- Simplify connectivity to a router from anywhere by securely connecting to the terminal server on a specific line.
- Allow modems attached to routers to be used for dial-out securely.
- Require authentication of each of the lines through a locally defined username and password, TACACS+, or RADIUS.

Note The **session slot** command that is used to start a session with a module requires Telnet to be accepted on the virtual tty (vty) lines. When you restrict vty lines only to SSH, you cannot use the command to communicate with the modules. This applies to any Cisco IOS device where the user can telnet to a module on the device.

How to Configure SSH Terminal-Line Access

Configuring SSH Terminal-Line Access

Perform this task to configure a Cisco router to support reverse secure Telnet.

Note SSH must already be configured on the router.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **line** *line-number* [*ending-line-number*]
4. **no exec**
5. **login** {**local** | **authentication** *listname*}
6. **rotary** *group*
7. **transport input** {**all** | **ssh**}
8. **exit**
9. **ip ssh port** *portnum* **rotary** *group*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**line** *line-number* [*ending-line-number*] **Example:** `Router(config)# line 1 200`	Identifies a line for configuration and enters line configuration mode. **Note** For router console configurations, each line must be defined in its own rotary, and SSH must be configured to listen in on each rotary. **Note** An authentication method requiring a username and password must be configured for each line. This may be done through the use of a local username and password stored on the router, through the use of TACACS+, or through the use of RADIUS. Neither Line passwords nor the enable password are sufficient to be used with SSH.
Step 4	**no exec** **Example:** `Router(config-line)# no exec`	Disables exec processing on each of the lines.

Step 5	**login** {**local** | **authentication** *listname*}
	
	Example:
	
	`Router(config-line)# login authentication default`
		Defines a login authentication mechanism for the lines.
		
		Note The authentication method must utilize a username and password.

Step 6	**rotary** *group* **Example:** `Router(config-line)# rotary 1`	Defines a group of lines consisting of one or more lines. **Note** All rotaries used must be defined, and each defined rotary must be used when SSH is enabled.

Step 7	**transport input** {**all** | **ssh**}
	
	Example:
	
	`Router(config-line)# transport input ssh`
		Defines which protocols to use to connect to a specific line of the router.

Step 8	**exit** **Example:** `Router(config-line)# exit`	Exits line configuration mode.
Step 9	**ip ssh port** *portnum* **rotary** *group* **Example:** `Router(config)# ip ssh port 2000 rotary 1`	Enables secure network access to the tty lines. • Use this command to connect the *portnum* argument with the rotary *group*argument, which is associated with a line or group of lines. **Note** The *group* argument must correspond with the **rotary** *group* number chosen in Step 6.

Verifying SSH Terminal-Line Access

To verify that this functionality is working, you can connect to a router using an SSH client.

Configuration Examples for SSH Terminal-Line Access

Example SSH Terminal-Line Access Configuration

The following example shows how to configure the SSH Terminal-Line Access feature on a modem used for dial-out on lines 1 through 200. To get any of the dial-out modems, use any SSH client and start an SSH session to port 2000 of the router to get to the next available modem from the rotary.

```
line 1 200
 no exec
 login authentication default
 rotary 1
 transport input ssh
 exit
ip ssh port 2000 rotary 1
```

Example SSH Terminal-Line Access for a Console Serial Line Ports Configuration

The following example shows how to configure the SSH Terminal-Line Access feature to access the console or serial line interface of various devices. For this type of access, each line is put into its own rotary, and each rotary is used for a single port. In this example, lines 1 through 3 are used; the port (line) mappings of the configuration are shown in the table below.

Table 6: Port (line) Configuration Mappings

Line Number	SSH Port Number
1	2001
2	2002
3	2003

```
line 1
 no exec
 login authentication default
 rotary 1
 transport input ssh
line 2
```

```
 no exec
 login authentication default
 rotary 2
 transport input ssh
line 3
 no exec
 login authentication default
 rotary 3
 transport input ssh
ip ssh port 2001 rotary 1 3
```

Additional References

Related Documents

Related Topic	**Document Title**
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
SSH	*Cisco IOS Security Configuration Guide: Securing User Services*
SSH commands	*Cisco IOS Security Command Reference*
Dial Technologies	*Cisco IOS Dial Technologies Configuration Guide*
Dial commands	*Cisco IOS Dial Technologies Command Reference*
Downloading a software image	*Cisco IOS Configuration Fundamentals Configuration Guide*

Standards

Standard	**Title**
	--

MIBs

MIB	**MIBs Link**
	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
None.	--

Technical Assistance

Description	Link
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.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for SSH Terminal-Line Access

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 7: Feature Information for SSH Terminal-Line Access

Feature Name	Releases	Feature Information
SSH Terminal-Line Access	12.2(4)JA 12.2(15)T 12.2(6th)S	The SSH Terminal-Line Access feature provides users secure access to tty (text telephone) lines. tty allows the hearing- and speech-impaired to communicate by using a telephone to type messages. This feature was introduced in Cisco IOS Release 12.2(4)JA. This feature was integrated into Cisco IOS Release 12.2(15)T. This feature was integrated into Cisco IOS Release 12.2(6th)S. The following command was introduced or modified: **ip ssh port**.

CHAPTER 7

AES-CTR Support for SSHv2

The AES-CTR Support for SSHv2 feature provides increased security through support for the Advanced Encryption Standard counter (AES-CTR) encryption mode during an encrypted Secure Shell version 2 (SSHv2) session between the server and the client.

Finding Feature Information

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

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

Prerequisites for AES-CTR Support for SSHv2

- Ensure that you use a Secure Shell (SSH) remote device that supports SSH Version 2 (SSHv2) and connect to a Cisco device.
- Ensure that both the client and the server that are used in the SSH session support the Advanced Encryption Standard counter mode (AES-CTR) encryption mode.

Restrictions for AES-CTR Support for SSHv2

- The Secure Shell (SSH) server and SSH client are supported only on crypto k9 (Triple Data Encryption Standard [3DES]) software images depending on your release.

Information About AES-CTR Support for SSHv2

Secure Shell Version 2 Encryption Modes

The Cisco Secure Shell (SSH) implementation enables a secure, encrypted connection between a server and client. The SSH servers and clients use the SSH protocol to provide device authentication and encryption.

To start an encrypted session between the SSH client and server, the preferred mode of encryption needs to be decided. For increased security, the preferred crypto algorithm for the SSH session is the Advanced Encryption Standard counter mode (AES-CTR).

SSH version 2 (SSHv2) supports AES-CTR encryption for 128-, 192-, and 256-bit key length. From the supported AES-CTR algorithms, the preferred algorithm is chosen based on the processing capability. The greater the length of the key, the stronger the encryption.

The Cisco SSH servers and clients support three types of crypto algorithms to encrypt data and selects the encryption mode in the following order of preferred encryption:

- AES-CTR
- AES Cipher Block Chaining (AES-CBC)
- Triple Data Encryption Standard (3DES)

If the SSH session uses a remote device that does not support the AES-CTR encryption mode, then the encryption mode for the session falls back to AES-CBC mode.

How to Configure AES-CTR Support for SSHv2

Starting an Encrypted Session from the SSH Client

Perform this task to start an encrypted Secure Shell (SSH) session from the SSH client using the Advanced Encryption Standard counter mode (AES-CTR) encryption mode.

Note The device with which you want to connect must support an SSH server that has the AES-CTR encryption algorithm that is supported in Cisco software. SSH can be run even when the device is disabled.

SUMMARY STEPS

1. **enable**
2. **ssh** [**-v** {**1** | **2**} | **-c** {**aes128-ctr** | **aes192-ctr** | **aes256-ctr** | **aes128-cbc** | **3des** | **aes192-cbc** | **aes256-cbc**} | **-l** *user-id* | **-l** *user-id***:***vrf-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**]
3. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**ssh** [**-v** {**1** | **2**} | **-c** {**aes128-ctr** | **aes192-ctr** | **aes256-ctr** | **aes128-cbc** | **3des** | **aes192-cbc** | **aes256-cbc**} | **-l** *user-id* | **-l** *user-id***:***vrf-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:** `Device# ssh -v 2 -c aes256-ctr -m hmac-sha1-96 -l user2 10.76.82.24`	Starts an encrypted session with a remote networking device.
Step 3	**exit** **Example:** `Device# exit`	Exits privileged EXEC mode.

Verifying the Encryption Mode Used in the SSH Server or Client

SUMMARY STEPS

1. **enable**
2. **show ssh**
3. **debug ip ssh detail**

DETAILED STEPS

Step 1 **enable**

Enables privileged EXEC mode.

- Enter your password if prompted.

Example:

```
Device> enable
```

Step 2 **show ssh**

Displays the encryption algorithms used for an encrypted session.

Example:

The following sample output from the **show ssh** command shows that the AES-CTR encryption mode is used for the session between the SSH server and client:

```
Device# show ssh

Connection Version Mode Encryption  Hmac                State         Username

0          1.99     IN   aes128-ctr  hmac-sha1     Session started       cisco
0          1.99     OUT  aes128-ctr  hmac-sha1     Session started       cisco

%No SSHv1 server connections running.
```

Step 3 **debug ip ssh detail**

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

Example:

The following sample output from the **debug ip ssh detail** command in the SSH server shows that the AES-CTR encryption mode is used for the session between the SSH server and client:

```
Device# debug ip ssh detail

SSH2 0: kex: client->server enc:aes128-ctr mac:hmac-md5
SSH2 0: kex: server->client enc:aes128-ctr mac:hmac-md5
```

The following sample output from the **debug ip ssh detail** command in the SSH client shows that the AES-CTR encryption mode is used for the session between the SSH server and client:

```
Device# debug ip ssh detail

SSH2 CLIENT 0: kex: server->client enc:aes128-ctr mac:hmac-md5
SSH2 CLIENT 0: kex: client->server enc:aes128-ctr mac:hmac-md5
```

Configuration Examples for AES-CTR Support for SSHv2

Example: Starting an Encrypted Session from the SSH Client

The following example shows how to start an encrypted Secure Shell (SSH) session from the SSH client using the Advanced Encryption Standard counter mode (AES-CTR) encryption mode:

```
Device> enable
Device# ssh -v 2 -c aes256-ctr -m hmac-sha1-96 -l user2 10.76.82.24
Device# exit
```

Additional References for AES-CTR Support for SSHv2

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
Security commands	• Cisco IOS Security Command Reference: Commands A to C • Cisco IOS Security Command Reference: Commands D to L • Cisco IOS Security Command Reference: Commands M to R • Cisco IOS Security Command Reference: Commands S to Z
SSH configuration	*Secure Shell Configuration Guide*

Standards and RFCs

Standard/RFC	Title
RFC 4344	*The Secure Shell (SSH) Transport Layer Encryption Modes*

Technical Assistance

Description	Link
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.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for AES-CTR Support for SSHv2

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 8: Feature Information for AES-CTR Support for SSHv2

Feature Name	Releases	Feature Information
AES-CTR Support for SSHv2	15.4(2)T 15.2(1)SY	The AES-CTR Support for SSHv2 feature provides increased security through support for the Advanced Encryption Standard counter (AES-CTR) encryption mode during an encrypted Secure Shell version 2 (SSHv2) session between the server and the client.

CHAPTER 8

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

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

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to 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** | **pasword**} 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** | **pasword**} 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** | **pasword**}
4. **ip ssh server authenticate user** {**publickey** | **keyboard** | **pasword**}
5. **default ip ssh server authenticate user**
6. **end**

DETAILED STEPS

	Command or Action	Purpose
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	**no ip ssh server authenticate user** {**publickey** | **keyboard** | **pasword**}
	
	Example:
	`Device(config)# no ip ssh server authenticate user publickey`
	`%SSH:Publickey disabled.Overriding RFC`
		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.

Step 4	**ip ssh server authenticate user** {**publickey** | **keyboard** | **pasword**} **Example:** `Device(config)# ip ssh server authenticate user publickey`	Enables the disabled user authentication method in the SSH server.
Step 5	**default ip ssh server authenticate user** **Example:** `Device(config)# default ip ssh server authenticate user`	Returns to the default behavior in which all user authentication methods are enabled in the predefined order.

	Command or Action	Purpose

Step 6	**end**
	
	Example:
	
	`Device(config)# end`
		Exits global configuration mode and returns to privileged EXEC mode.

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

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
Security commands	• Cisco IOS Security Command Reference: Commands A to C • Cisco IOS Security Command Reference: Commands D to L • Cisco IOS Security Command Reference: Commands M to R • Cisco IOS Security Command Reference: Commands S to Z
SSH configuration	*Secure Shell Configuration Guide*

Standards and RFCs

Standard/RFC	Title
RFC 4252	*The Secure Shell (SSH) Authentication Protocol*
RFC 4253	*The Secure Shell (SSH) Transport Layer Protocol*

Technical Assistance

Description	Link
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.	http://www.cisco.com/support

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

Feature Name	Releases	Feature Information
Secure Shell—Configuring User Authentication Methods	15.3(3)M 15.2(1)SY	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: **ip ssh server authenticate user**.