Secure Shell Configuration Guide Cisco IOS Release 12.2SR

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

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 the 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. For information about SSH Version 2, see the Secure Shell Version 2 Support feature module.

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

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see 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 document.

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

Perform the following tasks before configuring SSH:

- Download the required image on the router. The SSH server requires 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 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* for more information on downloading a software image.

- Configure a hostname and host domain for your router by using the **hostname** and **ip domain-name** commands in global configuration mode.
- Generate a Rivest, Shamir and Adleman (RSA) key pair for your router. 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 Configuring Authentication Configuring Authorization and Configuring Accounting feature modules for more information.

Restrictions for Configuring SSH

SSH has the following restrictions:

- The SSH server and SSH client are supported on 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

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

SSH Server

The SSH Server feature enables an SSH client to make a secure, encrypted connection to a Cisco router. 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 IOS software authentication. The SSH server in Cisco IOS software works with publicly and commercially available SSH clients.

SSH Integrated Client

The SSH Integrated Client feature is an application thats runs over the SSH protocol to provide device authentication and encryption. The SSH client enables a Cisco router to make a secure, encrypted

connection to another Cisco router or to any other device that is running the SSH server. This connection provides functionality that is 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 insecure network.

The SSH client in Cisco IOS software works with publicly and commercially available SSH servers. The SSH client supports the ciphers of DES, 3DES, and password authentication. User authentication is performed like that in the Telnet session to the router. 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

RSA authentication available in SSH clients is not supported on the SSH server for Cisco IOS software by default. See the " Configuring a Router for SSH Version 2 Using Private Public Key Pairs" section of the "Secure Shell Version 2 Support" chapter for the procedure to configure RSA authentication support.

How to Configure SSH

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

Configuring an SSH Server

Perform the following steps to configure an SSH server. This task helps you to enable the Cisco router for SSH.

Note The SSH client feature runs in user EXEC mode and has no specific configuration on the router.

Note The SSH commands are optional and are disabled when the SSH server is disabled. If SSH parameters are not configured, then the default values are used.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip ssh** {**timeout** *seconds* | **authentication-retries** *integer*}
4.

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.
Step 3	**ip ssh** {**timeout** *seconds* | **authentication-retries** *integer*} **Example:** `Router(config) # ip ssh timeout 30`	Configures SSH control parameters on your router. • Select one of the SSH control variables. • The *seconds* argument specifies the timeout in seconds, not to exceed 120 seconds. The default is 120. This setting applies to the SSH negotiation phase. Once the EXEC session starts, the standard timeouts configured for the vty apply. • By default, five vtys are defined (0-4); therefore five terminal sessions are possible. After the SSH executes a shell, the vty timeout starts. The vty timeout defaults to 10 minutes.
Step 4		• The *integer* argument specifies the number of authentication retries, not to exceed five authentication retries. The default is three. **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 router using the **ip ssh authentication-retries** *integer*command, plus one.

Invoking an SSH Client

Perform this task to invoke an SSH client.

SUMMARY STEPS

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

DETAILED STEPS

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

• Troubleshooting Tips, page 5

Troubleshooting Tips

- If your SSH configuration commands are rejected as illegal commands, you have not successfully generated an RSA key pair for your router. 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 router using the **hostname** global configuration command. See the IPsec and Quality of Service feature module for more information.

- ◦ No domain specified

You must configure a host domain for the router using the **ip domain-name** global configuration command. See the IPsec and Quality of Service feature module for more information.

- The number of allowable SSH connections is limited to the maximum number of vtys configured for the router. Each SSH connection uses a vty resource.
- SSH uses either local security or the security protocol that is configured through AAA on your router for user authentication. When configuring 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

This section provides the following configuration examples, which are output from the show running-config EXEC command on a Cisco 7200, Cisco 7500, and Cisco 12000 routers.

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

Note The **crypto key generate rsa** command is not displayed in the **show running-config** output.

Example SSH on a Cisco 7200 Series Router

In the following example, SSH is configured on a Cisco 7200 with a timeout that is not to exceed 60 seconds and no more than 2 authentication retries. Before the SSH server feature is configured on the router, TACACS+ is specified as the method of authentication.

```
hostname Router72K
aaa new-model
aaa authentication login default tacacs+
aaa authentication login aaa7200kw none
enable password password
username username1 password 0 password1
username username2 password 0 password2
ip subnet-zero
no ip domain-lookup
ip domain-name cisco.com
! Enter the ssh commands.
ip ssh timeout 60
ip ssh authentication-retries 2
controller E1 2/0
controller E1 2/1
interface Ethernet1/0
ip address 192.168.110.2 255.255.255.0 secondary
ip address 192.168.109.2 255.255.255.0
no ip directed-broadcast
no ip route-cache
no ip mroute-cache
no keepalive
no cdp enable
interface Ethernet1/1
no ip address
no ip directed-broadcast
no ip route-cache
no ip mroute-cache
shutdown
no cdp enable
interface Ethernet1/2
no ip address
no ip directed-broadcast
no ip route-cache
no ip mroute-cache
shutdown
no cdp enable
no ip classless
ip route 192.168.1.0 255.255.255.0 10.1.10.1
ip route 192.168.9.0 255.255.255.0 10.1.1.1
ip route 192.168.10.0 255.255.255.0 10.1.1.1
map-list atm
ip 10.1.10.1 atm-vc 7 broadcast
no cdp run
tacacs-server host 192.168.109.216 port 9000
```

```
tacacs-server key cisco
radius-server host 192.168.109.216 auth-port 1650 acct-port 1651
radius-server key cisco
line con 0
exec-timeout 0 0
login authentication aaa7200kw
transport input none
line aux 0
line vty 0 4
password password
end
```

Example SSH on a Cisco 7500 Series Router

In the following example, SSH is configured on a Cisco 7500 with a timeout that is not to exceed 60 seconds and no more than 5 authentication retries. Before the SSH server feature is configured on the router, RADIUS is specified as the method of authentication.

```
hostname Router75K
aaa new-model
aaa authentication login default radius
aaa authentication login aaa7500kw none
enable password password

username username1 password 0 password1
username username2 password 0 password2
ip subnet-zero
no ip cef
no ip domain-lookup
ip domain-name cisco.com
! Enter ssh commands.
ip ssh timeout 60
ip ssh authentication-retries 5
controller E1 3/0
channel-group 0 timeslots 1

controller E1 3/1
channel-group 0 timeslots 1
channel-group 1 timeslots 2
interface Ethernet0/0/0
no ip address
no ip directed-broadcast
no ip route-cache distributed
shutdown

interface Ethernet0/0/1
no ip address
no ip directed-broadcast
no ip route-cache distributed
shutdown
interface Ethernet0/0/2
no ip address
no ip directed-broadcast
no ip route-cache distributed
shutdown

interface Ethernet0/0/3
no ip address
no ip directed-broadcast
no ip route-cache distributed
shutdown

interface Ethernet1/0
ip address 192.168.110.2 255.255.255.0 secondary
ip address 192.168.109.2 255.255.255.0
no ip directed-broadcast
no ip route-cache
no ip mroute-cache

interface Ethernet1/1
```

```
ip address 192.168.109.2 255.255.255.0
no ip directed-broadcast
no ip route-cache
no ip mroute-cache
shutdown

interface Ethernet1/2
no ip address
no ip directed-broadcast
no ip route-cache
no ip mroute-cache

interface Ethernet1/3
no ip address
no ip directed-broadcast
no ip route-cache
no ip mroute-cache
shutdown

interface Ethernet1/4
no ip address
no ip directed-broadcast
no ip route-cache
no ip mroute-cache
shutdown
interface Ethernet1/5
no ip address
no ip directed-broadcast
no ip route-cache
no ip mroute-cache
shutdown

interface Serial2/0
ip address 10.1.1.2 255.0.0.0
no ip directed-broadcast
encapsulation ppp
no ip route-cache
no ip mroute-cache
ip classless
ip route 192.168.9.0 255.255.255.0 10.1.1.1
ip route 192.168.10.0 255.255.255.0 10.1.1.1
tacacs-server host 192.168.109.216 port 9000
tacacs-server key cisco
radius-server host 192.168.109.216 auth-port 1650 acct-port 1651
radius-server key cisco
line con 0
exec-timeout 0 0
login authentication aaa7500kw
transport input none
line aux 0
transport input all
line vty 0 4

end
```

Example SSH on a Cisco 12000 Series Router

In the following example, SSH is configured on a Cisco 12000 with a timeout that is not to exceed 60 seconds and no more than two authentication retries. Before the SSH server feature is configured on the router, TACACS+ is specified as the method of authentication.

```
hostname Router12K
aaa new-model
aaa authentication login default tacacs+ local
aaa authentication login aaa12000kw local
enable password password

username username1 password 0 password1
username username2 password 0 password2
redundancy
```

```
main-cpu
auto-sync startup-config
ip subnet-zero
no ip domain-lookup
ip domain-name cisco.com
! Enter ssh commands.
ip ssh timeout 60
ip ssh authentication-retries 2
interface ATM0/0
no ip address
no ip directed-broadcast
no ip route-cache cef
shutdown
interface POS1/0
ip address 10.100.100.2 255.255.255.0
no ip directed-broadcast
encapsulation ppp
no ip route-cache cef
no keepalive
crc 16
no cdp enable

interface POS1/1
no ip address
no ip directed-broadcast
no ip route-cache cef
shutdown
crc 32

interface POS1/2
no ip address
no ip directed-broadcast
no ip route-cache cef
shutdown
crc 32

interface POS1/3
no ip address
no ip directed-broadcast
no ip route-cache cef
shutdown
crc 32

interface POS2/0
ip address 10.1.1.1 255.255.255.0
no ip directed-broadcast
encapsulation ppp
no ip route-cache cef
crc 16

interface Ethernet0
ip address 172.17.110.91 255.255.255.224
no ip directed-broadcast

router ospf 1
network 0.0.0.0 255.255.255.255 area 0.0.0.0
ip classless
ip route 0.0.0.0 0.0.0.0 172.17.110.65

logging trap debugging
tacacs-server host 172.17.116.138
tacacs-server key cisco

radius-server host 172.17.116.138 auth-port 1650 acct-port 1651
radius-server key cisco

line con 0
exec-timeout 0 0
login authentication aaa12000kw
transport input none
line aux 0
line vty 0 4
```

```
no scheduler max-task-time
no exception linecard slot 0 sqe-registers
no exception linecard slot 1 sqe-registers
no exception linecard slot 2 sqe-registers
no exception linecard slot 3 sqe-registers
no exception linecard slot 4 sqe-registers
no exception linecard slot 5 sqe-registers
no exception linecard slot 6 sqe-registers
end
```

Example Verifying SSH

To verify that the 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:

```
Router# show ip ssh
SSH Enabled - version 1.5
Authentication timeout: 120 secs; Authentication retries: 3
```

The following example shows that SSH is disabled:

```
Router# 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 router when SSH is enabled:

```
Router# show ssh
Connection      Version     Encryption    State    Username
    0    1.5    3DES    Session Started        guest
```

The following example shows that SSH is disabled:

```
Router# show ssh
%No SSH server connections running.
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Authentication, authorization, and accounting (AAA)	• Configuring Accounting feature module • Configuring Authentication feature module • Configuring Authorization feature module
IPsec	IPsec and Quality of Service feature module
SSH Version 2	Secure Shell Version 2 Support feature module
Downloading a software image	*Cisco IOS Configuration Fundamentals Configuration Guide*

Standards

Standard	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	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
No new or modified RFCs are supported by this feature.	--

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	The Secure Shell (SSH) feature is an application and a protocol that provides a secure replacement to the Berkeley r-tools. The protocol secures the sessions using standard cryptographic mechanisms, and the application can be used similarly to the Berkeley rexec and rsh tools. Two versions of SSH are available: SSH Version 1 and SSH Version 2. This document describes SSH Version 1.

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 feature information and caveats, see 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 document.

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

Cisco IOS software has for quite some time included a feature called Reverse telnet, whereby you can telnet to a certain port range and connect to terminal or auxiliary lines. Reverse telnet has often been used to connect a Cisco IOS router that has many terminal lines to the consoles of other Cisco IOS routers or to other devices. Telnet makes it easy to reach the router console from anywhere simply by telnet to the terminal server on a specific line. This telnet approach can be used to configure a router even if all network connectivity to that router is disconnected. Reverse telnet also allows modems that are attached to Cisco IOS routers 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, page 14."

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:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**line** *line-number ending-line-number* **Example:** `Router# line 1 3`	Identifies a line for configuration and enters line configuration mode.
Step 4	**no exec** **Example:** `Router (config-line)# no exec`	Disables EXEC processing on a line.
Step 5	**login authentication** *listname* **Example:** `Router (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:** `Router (config-line)# transport input ssh`	Defines which protocols to use to connect to a specific line of the router. • The **ssh** keyword must be used for the Reverse SSH Enhancements feature.
Step 7	**exit** **Example:** `Router (config-line)# exit`	Exits line configuration mode.
Step 8	**exit** **Example:** `Router (config)# exit`	Exits global configuration mode.
Step 9	**ssh -l** *userid* **:** {*number*} {*ip-address*} **Example:** `Router# 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:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**line** *line-number ending-line-number* **Example:** `Router# line 1 200`	Identifies a line for configuration and enters line configuration mode.
Step 4	**no exec** **Example:** `Router (config-line)# no exec`	Disables EXEC processing on a line.
Step 5	**login authentication** *listname* **Example:** `Router (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:** `Router (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:** `Router (config-line)# transport input ssh`	Defines which protocols to use to connect to a specific line of the router. • The **ssh** keyword must be used for the Reverse SSH Enhancements feature.
Step 8	**exit** **Example:** `Router (config-line)# exit`	Exits line configuration mode.
Step 9	**exit** **Example:** `Router (config)# exit`	Exits global configuration mode.
Step 10	**ssh -l** *userid* **:rotary** {*number*} {*ip-address*} **Example:** `Router# 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:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**debug ip ssh client** **Example:** `Router# 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:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**debug ip ssh** **Example:** `Router# debug ip ssh`	Displays debugging messages for the SSH server.

	Command or Action	Purpose
Step 3	**show ssh** **Example:** `Router# show ssh`	Displays the status of the SSH server connections.
Step 4	**show line** **Example:** `Router# 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	See the following modules: • Configuring Secure Shell • Secure Shell Version 2 Support • SSH Terminal-Line Access
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	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. This feature was introduced in Cisco IOS Release 12.3(11)T. The following command was introduced: **ssh**.

Secure Copy

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

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see 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 document.

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

Information About Secure Copy

How Secure Copy Works

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

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

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

How to Configure Secure Copy

- Configuring Secure Copy, page 26

Configuring Secure Copy

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

SUMMARY STEPS

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

DETAILED STEPS

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

Command or Action	Purpose
Step 2 **configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3 **aaa new-model** **Example:** `Router(config)# aaa new-model`	Sets AAA authentication at login.
Step 4 **aaa authentication login** {**default** | *list-name*} *method1*[*method2...*] **Example:** `Router(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:** `Router(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 it when you configure SCP.
Step 6 **username** *name* [**privilege** *level*] {**password** *encryption-type encrypted-password*} **Example:** `Router(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:** `Router(config)# ip scp server enable`	Enables SCP server-side functionality.

	Command or Action	Purpose
Step 8	**show running-config** **Example:** `Router# show running-config`	(Optional) Verifies the SCP server-side functionality.
Step 9	**debug ip scp** **Example:** `Router# debug ip scp`	(Optional) Troubleshoots SCP authentication problems.

Configuration Examples for Secure Copy

Example SCP Server-Side Configuration Using Local Authentication

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

```
! AAA authentication and authorization must be configured properly for SCP to work.
aaa new-model
aaa authentication login default local
aaa authorization exec default local
username user1 privilege 15 password 0 lab
! SSH must be configured and functioning properly.
ip ssh time-out 120
ip ssh authentication-retries 3
ip scp server enable
```

Example SCP Server-Side Configuration Using Network-Based Authentication

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

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

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Secure Shell Version 1 and 2 support	• Configuring Secure Shell module • Secure Shell Version 2 Support module
Authentication and authorization commands	Cisco IOS Security Command Reference
Configuring authentication and authorization	Authentication, Authorization, and Accounting (AAA) section of *Cisco IOS Security Configuration Guide: Securing User Services* , Release 15.0

Standards

Standards	Title
None	--

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 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	12.2(2)T 12.0(21)S 12.2(25)S	The Secure Copy (SCP) feature provides a secure and authenticated method for copying router configuration or router image files. SCP relies on Secure Shell (SSH), an application and a protocol that provide a secure replacement for the Berkeley r-tools. This feature was introduced in Cisco IOS Release 12.2(2)T. This feature was integrated into Cisco IOS Release 12.0(21)S. This feature was integrated into Cisco IOS Release 12.2(25)S. The following commands were introduced or modified: **debug ip scp, ip scp server enable**.

Glossary

AAA --authentication, authorization, and accounting. Framework of security services that provide the method for identifying users (authentication), for remote access control (authorization), and for collecting and sending security server information used for billing, auditing, and reporting (accounting).

rcp --remote copy. Relying on Remote Shell (Berkeley r-tools suite) for security, rcp copies files, such as router images and startup configurations, to and from routers.

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

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

Secure Shell Version 2 Support

The Secure Shell Version 2 Support feature allows you to configure Secure Shell (SSH) Version 2 (SSH Version 1 support was implemented in an earlier Cisco IOS 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 feature information and caveats, see 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 document.

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

Prior to configuring SSH, ensure that the required image is loaded on your router. The SSH server requires you to have a k9 (Triple Data Encryption Standard [3DES]) software image from Cisco IOS Release 12.3(4)T, 12.2(25)S, or 12.3(7)JA downloaded on to your router.

Note The SSH Version 2 server is supported in Cisco IOS Release 12.3(4)T, 12.3(2)XE, 12.2(25)S, and 12.3(7)JA; the SSH Version 2 client is supported beginning with Cisco IOS Release 12.3(7)T and is supported in Cisco IOS Release12.3(7)JA. (The SSH client runs both the SSH Version 1 protocol and the Version 2 protocol and is supported in both k8 and k9 images in Cisco IOS Release 12.3(4)T.)

For more information about downloading a software image, refer to *Cisco IOS Configuration Fundamentals Configuration Guide, Release 12.4T and Cisco IOS Network Management Configuration Guide, Release 15.0*.

Restrictions for Secure Shell Version 2 Support

- SSH servers and SSH clients are supported in 3DES software images.
- Execution Shell, remote command execution, and SCP are the only applications supported.
- Rivest, Shamir, and Adleman (RSA) key generation is an SSH server-side requirement. Routers that act as SSH clients need not generate RSA keys.
- The RSA key pair size must be greater than or equal to 768.
- The following functionality is 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 was introduced so that you may define which version of SSH to configure. 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 router 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 was also introduced in Cisco IOS Release 12.3(4)T so that you can enable an SSH connection using the 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). The behavior still exists, but by using the **ip ssh rsa keypair-name** command, you can overcome that 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 IOS 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 VRF-aware SSH, SSH debug enhancements, and Diffie-Hellman (DH) group exchange support.

Note Only the VRF-aware SSH feature is supported in Cisco IOS Release 12.2(50)SY.

The Cisco IOS 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 was introduced in Cisco IOS Release 12.4(20)T so that you can configure 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 allow you to simplify the debugging process. Previously, 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 IOS SSH Version 2 (SSHv2) supports keyboard-interactive and password-based authentication methods. The SSHv2 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 IOS SSH server to complete the authentication.

An SSH user trying to establish the 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 IOS 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 IOS SSH client must assign a host key for each server. When the client tries to establish an SSH session with a server, it 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 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 IOS server, but Cisco IOS clients cannot propose public key as an authentication method. If the Cisco IOS 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 IOS SSH client.

SNMP Trap Generation

Effective with Cisco IOS Release 12.4(17), 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 turned on. For information about enabling SNMP traps, see the Configuring SNMP Support module in the *Cisco IOS Network Management Configuration Guide, Release 15.0* .

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 Example Setting an SNMP Trap, page 54.”

You must also turn on 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, page 56.

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, page 54.

How to Configure Secure Shell Version 2 Support

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

Perform this task to configure a router for SSH Version 2 using a hostname and domain name. You may also configure SSH Version 2 by using the RSA key pair configuration (see the Configuring a Router for SSH Version 2 Using RSA Key Pairs, page 38).

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **hostname** *hostname*
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**]

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	**hostname** *hostname* **Example:** `Router(config)# hostname cisco 7200`	Configures a hostname for your router.
Step 4	**ip domain-name** *name* **Example:** `Router(config)# ip domain-name example.com`	Configures a domain name for your router.
Step 5	**crypto key generate rsa** **Example:** `Router(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:** `Router(config)# ip ssh time-out 120`	(Optional) Configures SSH control variables on your router.
Step 7	**ip ssh version** [**1** | **2**] **Example:** `Router(config)# ip ssh version 1`	(Optional) Specifies the version of SSH to be run on your router.

Configuring a Router for SSH Version 2 Using RSA Key Pairs

Perform this task to enable SSH Version 2 without configuring a hostname or a domain name. SSH Version 2 will be enabled if the key pair that you configure already exists or if it is generated later. You may also configure SSH Version 2 by using the hostname and domain name configuration (see the Configuring a Router for SSH Version 2 Using a Hostname and Domain Name, page 37).

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**

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.
Step 3	**ip ssh rsa keypair-name** *keypair-name* **Example:** `Router(config)# ip ssh rsa keypair-name sshkeys`	Specifies which RSA key pair to use for SSH usage. **Note** A Cisco IOS router can have many RSA key pairs.
Step 4	**crypto key generate rsa usage-keys label** *key-label* **modulus** *modulus-size* **Example:** `Router(config)# crypto key generate rsa usage-keys label sshkeys modulus 768`	Enables the SSH server for local and remote authentication on the router. • 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. After you have deleted the RSA key pair, you automatically disable the SSH server.
Step 5	**ip ssh** [**time-out** *seconds* | **authentication-retries** *integer*] **Example:** `Router(config)# ip ssh time-out 12`	Configures SSH control variables on your router.
Step 6	**ip ssh version 2** **Example:** `Router(config)# ip ssh version 2`	Specifies the version of SSH to be run on a router.

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

Perform this task to configure the Cisco IOS SSH server to perform RSA-based user authentication. The user authentication is successful if the RSA public key stored on the server is verified with the public or the private key pair stored on the client.

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. **exit**
10. **key-hash** *key-type key-name*
11. **end**

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.
Step 3	**hostname** *name* **Example:** `Router(config)# hostname host1`	Specifies the hostname.

	Command or Action	Purpose
Step 4	**ip domain-name** *name* **Example:** `Router(config)# ip domain-name name1`	Defines a default domain name that the Cisco IOS software uses to complete unqualified hostnames.
Step 5	**crypto key generate rsa** **Example:** `Router(config)# crypto key generate rsa`	Generates RSA key pairs.
Step 6	**ip ssh pubkey-chain** **Example:** `Router(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	**username** *username* **Example:** `Router(conf-ssh-pubkey)# username user1`	Configures the SSH username and enters public-key user configuration mode.
Step 8	**key-string** **Example:** `Router(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	**exit** **Example:** `Router(conf-ssh-pubkey-data)# exit`	Exits public-key user configuration mode.

	Command or Action	Purpose
Step 10	**key-hash** *key-type key-name* **Example:** `Router(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 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 pubkey string or you can also copy the hash value from another Cisco IOS router. 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:** `Router(conf-ssh-pubkey-data)# end`	Exits the current mode and returns to privileged EXEC mode.

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

Perform this task to configure 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:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**hostname** *name* **Example:** `Router(config)# hostname host1`	Specifies the hostname.
Step 4	**ip domain-name** *name* **Example:** `Router(config)# ip domain-name name1`	Defines a default domain name that the Cisco IOS software uses to complete unqualified hostnames.
Step 5	**crypto key generate rsa** **Example:** `Router(config)# crypto key generate rsa`	Generates RSA key pairs.
Step 6	**ip ssh pubkey-chain** **Example:** `Router(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:** `Router(conf-ssh-pubkey)# server server1`	Enables the SSH server for public-key authentication on the router and enters public-key server configuration mode.

	Command or Action	Purpose
Step 8	**key-string** **Example:** `Router(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:** `Router(conf-ssh-pubkey-data)# exit`	Exits public-key data configuration mode and enters public-key server configuration mode.
Step 10	**key-hash** *key-type key-name* **Example:** `Router(conf-ssh-pubkey-server)# key-hash ssh-rsa key1`	(Optional) Specifies the SSH key type and version. • The key type must be ssh-rsa for 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 IOS router. 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:** `Router(conf-ssh-pubkey-server)# end`	Exits public-key server mode and returns to privileged EXEC mode.
Step 12	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 13	**ip ssh stricthostkeycheck** **Example:** `Router(config)# ip ssh stricthostkeycheck`	Ensures that the server authentication takes place. • The connection is terminated on a failure.

Starting an Encrypted Session with a Remote Device

Perform this task to start an encrypted session with a remote networking device, (You need not enable your router. SSH can be run in disabled mode.)

Note The device you want to connect with must support an SSH server that has an encryption algorithm that is supported in Cisco IOS software.

SUMMARY STEPS

1. **ssh** [**-v** {**1** | **2**}][**-c** {**3des** | **aes128-cbc** | **aes192-cbc** | **aes256-cbc**}] [**-m** {**hmac-md5** | **hmac-md5-96** | **hmac-sha1** | **hmac-sha1-96**}] [**l** *userid*] [**-o numberofpasswordprompts** *n*] [**-p** *port-num*]{*ip-addr* | *hostname*} [*command*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ssh** [**-v** {**1** | **2**}][**-c** {**3des** | **aes128-cbc** | **aes192-cbc** | **aes256-cbc**}] [**-m** {**hmac-md5** | **hmac-md5-96** | **hmac-sha1** | **hmac-sha1-96**}] [**l** *userid*] [**-o numberofpasswordprompts** *n*] [**-p** *port-num*]{*ip-addr* | *hostname*} [*command*] **Example:** `Router# ssh -v 2 -c aes256-cbc -m hmac-sha1-96 -l user2 10.76.82.24`	Starts an encrypted session with a remote networking device.

- Troubleshooting Tips, page 45

Troubleshooting Tips

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

Enabling Secure Copy Protocol on the SSH Server

Perform this task to enable secure copy protocol on the SSH server. This task configures the server-side functionality for SCP. This task shows a typical configuration that allows the router to securely copy files from a remote workstation.

SCP relies on authentication, authorization, and accounting (AAA) to function correctly. Therefore, AAA must be configured on the router.

SUMMARY STEPS

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

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.
Step 3	**aaa new-model** **Example:** `Router(config)# aaa new-model`	Enables the AAA access control model.
Step 4	**aaa authentication login default local** **Example:** `Router(config)# aaa authentication login default local`	Sets AAA authentication at login to use the local username database for authentication.
Step 5	**aaa authorization exec default local** **Example:** `Router(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 uses the local database for authorization.

	Command or Action	Purpose
Step 6	**username** *name* **privilege** *privilege-level* **password** *password* **Example:** `Router(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:** `Router(config)# ip ssh time-out 120`	Sets the time interval (in seconds) that the router waits for the SSH client to respond.
Step 8	**ip ssh authentication-retries** *integer* **Example:** `Router(config)# ip ssh authentication-retries 3`	Sets the number of authentication attempts after which the interface is reset.
Step 9	**ip scp server enable** **Example:** `Router(config)# ip scp server enable`	Enables the router to securely copy files from a remote workstation.

- Troubleshooting Tips, page 47

Troubleshooting Tips

To troubleshoot SCP authentication problems, use the **debug ip scp**command.

Verifying the Status of the Secure Shell Connection Using the show ssh Command

To display the status of the SSH connection on your router, use the **show ssh** command.

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show ssh** **Example:** `Router# show ssh`	Displays the status of SSH server connections.

Examples

Version 1 and Version 2 Connections

Version 2 Connection with No Version 1

Version 1 Connection with No Version 2

The following sample output from the **show ssh** command display status about various SSH Version 1 and Version 2 connections.

```
-------------------------------------------------------------------------
Router# 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
-------------------------------------------------------------------------


-------------------------------------------------------------------------
Router# 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.
-------------------------------------------------------------------------


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

Verifying the Secure Shell Status

Perform this task to verify your SSH configuration.

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show ip ssh** **Example:** `Router# show ip ssh`	Displays the version and configuration data for SSH.

Examples

Version 1 and Version 2 Connections

Version 2 Connection with No Version 1

Version 1 Connection with No Version 2

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:

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


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


---------------------------------------------------------------------
Router# 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

To display debug messages about the SSH connections, use the **debug ip ssh** command and the **debug snmp packet** command.

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**debug ip ssh** **Example:** `Router# debug ip ssh`	Displays debugging messages for SSH.
Step 3	**debug snmp packet** **Example:** `Router# debug snmp packet`	Displays information about every SNMP packet sent or received by the router.

Example

The following sample output from the **debug ip ssh** command shows that the digit 2 keyword has been assigned, signifying that it is an SSH Version 2 connection:

```
Router# 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

The following example shows how to configure SSH Version 1:

```
Router# configure terminal
Router(config)# ip ssh version 1
Router(config)# end
```

Example ConfiguringSecureShellVersion2

The following example shows how to configure SSH Version 2:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# ip ssh version 2
Router(config)# end
```

Example Configuring Secure Shell Versions 1 and 2

The following example shows how to configure both SSH Version 1 and SSH Version 2:

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

Example Starting an Encrypted Session with a Remote Device

The following example shows how to start an encrypted session with a remote device:

```
Router# 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 router. This example uses a locally defined username and password.

```
Router# configure terminal
Router(config)# aaa new-model
Router(config)# aaa authentication login default local
Router(config)# aaa authorization exec default local
Router(config)# username samplename privilege 15 password password1
Router(config)# ip ssh time-out 120
Router(config)# ip ssh authentication-retries 3
Router(config)# ip scp server enable
Router(config)# end
```

Example Setting an SNMP Trap

The following example shows that an SNMP trap has been 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 section Example SNMP Debugging, page 56."

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

Examples SSH Keyboard Interactive Authentication

Client-Side Debugs

In the following example, client-side debugs are turned on and the maximum number of prompts = six, (three for the SSH Keyboard Interactive Authentication method and for the password method of authentication).

```
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]
Router1# debug ip ssh client
SSH Client debugging is on
Router1# 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
Router2>
*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
```

TACACS ACS Is the Back-end AAA Server ChPass Is Enabled and a Blank Password Change Is Made

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

```
Router1# ssh -l cisco 10.1.1.3
Password:
Old Password: cisco
New Password: cisco123
Re-enter New password: cisco123
Router2> exit
[Connection to 10.1.1.3 closed by foreign host]
```

TACACS ACS Is the Back-end AAA Server ChPass Is Enabled and the Password Is Changed on First Login

In the following example, a TACACS+ ACS is the back-end server and the ChPass feature is enabled. The password is changed on the first login using the SSH Keyboard Interactive Authentication method.

```
Router1# 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
Router2> exit
[Connection to 10.1.1.3 closed by foreign host]
Router1# 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
Router2>
```

TACACS ACS Is the Back-end AAA Server ChPass Is Enabled and the Password Expires After Three Logins

In the following example, a TACACS+ ACS is the back-end AAA server and the ChPass feature is enabled. The password expires after three logins using the SSH Keyboard Interactive Authentication method.

```
Router# ssh -l cisco. 10.1.1.3
Password: cisco
Router2> exit
[Connection to 10.1.1.3 closed by foreign host]
Router1# ssh -l cisco 10.1.1.3
Password: cisco
Router2> exit
Router1# ssh -l cisco 10.1.1.3
Password: cisco
Router2> exit
[Connection to 10.1.1.3 closed by foreign host]
Router1# 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
Router2>
```

Example SNMP Debugging

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

```
Router1# debug snmp packet
SNMP packet debugging is on
Router1# ssh -l lab 10.0.0.2
Password:
Router2# exit
[Connection to 10.0.0.2 closed by foreign host]
Router1#
*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
Router1#
```

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.

```
Router# 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.

```
Router# 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

Where to Go Next

You have to use a SSH remote device that supports SSH Version 2, and you have to connect to a Cisco IOS router.

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
AAA	*Cisco IOS Security Configuration Guide: Securing User Services*
• Configuring a hostname and host domain • Configuring Secure Shell	"Configuring Secure Shell" module in the *Cisco IOS Security Configuration Guide: Securing User Services* .
Debugging commands	*Cisco IOS Debug Command Reference*
Downloading a Cisco software image	*Cisco IOS Configuration Fundamentals Configuration Guide Cisco IOS Network Management Configuration Guide*
Cisco IOS configuration fundamentals	

Related Topic	Document Title
IPSec	*Cisco IOS Security Configuration Guide: Secure Connectivity*
Security commands	*Cisco IOS Security Command Reference*
SNMP, configuring traps	Configuring SNMP Support module in the *Cisco IOS Network Management Configuration Guide*

Standards

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

MIBs

MIBs	MIBs Link
No new or modified MIBs are supported and support for existing MIBs has not been modified.	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
No new or modified RFCs are supported and support for existing RFCs has not been modified.	--

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 4 Feature Information for Secure Shell Version 2 Support

Feature Name	Releases	Feature Information
Secure Shell Version 2 Support	12.2(25)S 12.3(4)T 12.2(11)T	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. In 12.3(11)T, support was added for the Cisco 10000 series router. 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	12.0(32)SY 12.3(7)JA 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	12.4(18) 12.2(33)SXH3	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	12.4(20)T 15.1(2)S 12.2(50)SY	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.	15.0(1)M 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**.

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 feature information and caveats, see 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 document.

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.

Command or Action	Purpose
Step 2 **configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
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.

	Command or Action	Purpose
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 5 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 6 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**.