Secure Shell Version 2 Support

First Published: November 3, 2003
Last Updated: March 31, 2011

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 for Secure Shell Version 2 Support" section on page 26.

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

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

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 Adelman (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.

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" section on page 20."

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" section on page 22.

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" section on page 20.

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" section on page 6).

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.
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" section on page 5).

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

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

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	**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.
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

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.

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

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

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 server-side functionality for SCP. This example shows a typical configuration that allows the router to securely copy files from a remote workstation.

Prerequisites

SCP relies on AAA authentication and authorization 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

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.
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

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

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

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

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
-----------------------------------------------------------------------
```

Version 2 Connection with No Version 1

```
-----------------------------------------------------------------------
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.
-----------------------------------------------------------------------
```

Version 1 Connection with No Version 2

```
-----------------------------------------------------------------------
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

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

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:

Version 1 and Version 2 Connections

```
---------------------------------------------------------------------------
Router# show ip ssh

SSH Enabled - version 1.99
Authentication timeout: 120 secs; Authentication retries: 3
---------------------------------------------------------------------------
```

Version 2 Connection with No Version 1

```
---------------------------------------------------------------------------
Router# show ip ssh

SSH Enabled - version 2.0
Authentication timeout: 120 secs; Authentication retries: 3
---------------------------------------------------------------------------
```

Version 1 Connection with No Version 2

```
---------------------------------------------------------------------------
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

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.

Examples

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: Configuring Secure Shell Version 2

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" section on page 22."

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

Examples: SSH Keyboard Interactive Authentication

The following are examples of various scenarios in which the SSH Keyboard Interactive Authentication feature has been automatically deployed.

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	
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

Table 1 lists the features in this module and provides links to specific configuration information.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 1 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.

Table 1 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 sections provide information about this feature: • Information About Secure Shell Version 2 Support, page 2 • How to Configure Secure Shell Version 2 Support, page 5 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. The following sections provide information about this feature: • SNMP Trap Generation, page 4 • Example: SNMP Debugging, page 22
SSH Keyboard Interactive Authentication	12.4(18) 12.2(33)SXH3	This 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 following sections provide information about this feature: • SSH Keyboard Interactive Authentication, page 5 • Examples: SSH Keyboard Interactive Authentication, page 20

Table 1 Feature Information for Secure Shell Version 2 Support (continued)

Feature Name	Releases	Feature Information
Secure Shell Version 2 Enhancements	12.4(20)T 15.1(2)S	The Secure Shell Version 2 Enhancements feature includes a number of additional capabilities such as support for VRF aware SSH, SSH debug enhancements, and DH Group 14 and Group 16 exchange support. In Cisco IOS 15.1(2)S, support was added for the Cisco 7600 series router. The following sections provide information about this feature: • Secure Shell Version 2 Enhancements, page 3 • Example: Configuring Server-Side SCP, page 20 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 sections provide information about this feature: • Secure Shell Version 2 Enhancements for RSA Keys, page 3 • Configuring the Cisco IOS SSH Server to Perform RSA-Based User Authentication, page 8 The following commands were introduced or modified: **ip ssh pubkey-chain**, **ip ssh stricthostkeycheck**.