Secure Shell

Feature History for Secure Shell

This table provides release and platform support information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature Name and Description	Supported Platform
Cisco IOS XE 17.18.1	SSH: SSH is an application and protocol that provides a secure, encrypted replacement for less secure remote connection tools like Telnet and the Berkeley r-tools.	Cisco C9350 Series Smart Switches Cisco C9610 Series Smart Switches

What is Secure Shell?

Secure Shell (SSH) is an application and protocol that provides a secure, encrypted replacement for less secure remote connection tools like Telnet and the Berkeley r-tools.

It secures sessions using

- standard cryptographic mechanisms,
- provides device authentication and encryption,

- and allows for secure communication over an unsecured network.

Supported authentication methods for SSH include:

- TACACS+
- RADIUS
- Local authentication and authorization

To provide secure, authenticated, and encrypted remote access, SSH implementations include features such as primary applications, mechanisms, encryption algorithms, and protocol extensions. The following sections describe these features.

Features of SSH

SSH Server

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

An SSH server is a feature that enables an SSH client to make a secure, encrypted connection to a Cisco device. It provides functionality similar to an inbound Telnet connection but allows strong encryption to be used with Cisco software authentication. The SSH server in Cisco software works with publicly and commercially available SSH clients.

SSH Integrated Client

An SSH integrated client is an application that runs over the SSH protocol to provide device authentication and encryption. It enables a Cisco device to make a secure, encrypted connection to another device running an SSH server, providing functionality similar to an outbound Telnet connection but encrypted. The SSH client in Cisco software works with publicly and commercially available SSH servers.

Supported ciphers for the SSH client include:

- Data Encryption Standard (DES)
- 3DES

The SSH client supports password authentication.

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

The SSH server feature allows an SSH client to connect securely, while the SSH integrated client feature allows a device to initiate a secure connection to another SSH server. Both work together to ensure secure communication over potentially insecure networks.

RSA Authentication Support

Rivest, Shamir, and Adleman (RSA) authentication available in Secure Shell (SSH) clients is not supported on the SSH server for Cisco software by default.

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

The SSH server and SSH integrated client are applications that run on the switch. The SSH server works with the SSH client supported in this release and with non-Cisco SSH clients. The SSH client works with publicly and commercially available SSH servers. The SSH client supports the ciphers of Data Encryption Standard (DES), 3DES, and password authentication.

User authentication is performed like that in the Telnet session to the device. SSH also supports the following user authentication methods:

- TACACS+
- RADIUS
- Local authentication and authorization

SSH Configuration Guidelines

- RSA key pairs generated by SSHv1 servers can be used by SSHv2 servers, and vice versa.
- If the SSH server is running on an active switch and the active switch fails, the new active switch uses the RSA key pair generated by the previous active switch.
- If you receive CLI error messages after entering the crypto key generate **rsa** command, reconfigure the hostname and domain, and then enter the command again.
- If the message "No host name specified" appears when generating the RSA key pair, configure a hostname using the hostname command in global configuration mode.
- If the message "No domain specified" appears when generating the RSA key pair, configure an IP domain name using the **ip domain name** command in global configuration mode.
- Ensure that AAA is disabled on the console when configuring local authentication and authorization.

Strict-KEX

Strict-KEX is a protocol extension that strengthens security by ensuring that unexpected or out-of-sequence packets result in immediate connection termination and by resetting sequence numbers after key exchanges. To provide continuous protection, strict-KEX is enabled by default and cannot be disabled.

Prerequisites for Configuring Secure Shell

Ensure that all device and software prerequisites are met before configuring Secure Shell (SSH) to guarantee secure and reliable operation.

General SSH Prerequisites

Verify the following requirements before enabling SSH or Secure Copy Protocol (SCP) on your device:

- For SSH to work, the switch needs a Rivest, Shamir, and Adleman (RSA) public/private key pair. This is the same with Secure Copy Protocol (SCP), which relies on SSH for its secure transport.
- Download the required image on the device. The Secure Shell (SSH) server requires an IPsec (Data Encryption Standard [DES] or 3DES) encryption software image; the SSH client requires an IPsec (DES or 3DES) encryption software image.
- Configure a hostname and host domain for your device by using the **hostname** and **ip domain name** commands in global configuration mode.
- Generate a Rivest, Shamir, and Adleman (RSA) key pair for your device. This key pair automatically enables SSH and remote authentication when the **crypto key generate rsa** command is entered in privileged EXEC mode.

Authentication and Compatibility Requirements

Meet the following authentication and compatibility requirements to ensure proper SSH operation:

- Configure user authentication for local or remote access. You can configure authentication with or without authentication, authorization, and accounting (AAA).
- The Secure Shell (SSH) server requires an IPsec (Data Encryption Standard [DES] or 3DES) encryption software image; the SSH client requires an IPsec (DES or 3DES) encryption software image.
- For Strict-KEX to function, devices must be running Cisco IOS-XE with SSH support enabled.
- Both client and server devices should be compatible with the strict-kex algorithms for Strict-KEX to be effective.

Important Notes

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

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.

Restrictions for Configuring Secure Shell

The Secure Shell (SSH) server and SSH client are supported only on Data Encryption Standard (DES) (56-bit) and 3DES (168-bit) data encryption software images. 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.
- The SFTP server is not supported.

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

Configure Secure Shell

Learn how to configure Secure Shell (SSH) to enable secure remote access to your device and protect management sessions from unauthorized access.

Set Up the Device to Run SSH

Set up your device to run SSH by configuring authentication, hostname, domain name, and generating RSA keys.

This procedure enables secure remote access to your device using SSH by ensuring all necessary configurations are in place.

Before you begin

Configure user authentication for local or remote access. This step is required.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

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

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **hostname** *hostname*

Example:

```
Device(config)# hostname your_hostname
```

Configures a hostname and IP domain name for your device.

Note
Follow this procedure only if you are configuring the device as an SSH server.

Step 4 **ip domain name** *domain_name*

Example:

```
Device(config)# ip domain name your_domain
```

Configures a host domain for your device.

Step 5 **exit**

Example:

```
Device(config)# exit
```

Exits global configuration mode and returns to privileged EXEC mode.

Step 6 **crypto key generate rsa**

Example:

```
Device# crypto key generate rsa
```

Enables the SSH server for local and remote authentication on the device and generates an RSA key pair. Generating an RSA key pair for the device automatically enables SSH.

We recommend that a minimum modulus size of 1024 bits.

When you generate RSA keys, you are prompted to enter a modulus length. A longer modulus length might be more secure, but it takes longer to generate and to use.

Note
Follow this procedure only if you are configuring the device as an SSH server.

Step 7 **show ip ssh**

Example:

```
Device# show ip ssh
```

(Optional) Verifies that the SSH server is enabled and displays the version and configuration data for the SSH connection.

Configure an SSH Server

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

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

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

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **ip ssh** { **time-out** *seconds* | **authentication-retries** *integer* }

Example:

```
Device(config)# ip ssh time-out 30
```

Configures Secure Shell (SSH) control parameters.

Note
Note This command can also be used to establish the number of password prompts provided to the user. The number is the lower of the following two values:

- Value proposed by the client using the **ssh -o numberofpasswordprompt** command.
- Value configured on the device using the **ip ssh authentication-retries** *integer* command, plus one.

Step 4 **ip ssh rekey** { **time** *time* | **volume** *volume* }

Example:

```
Device(config)# ip ssh rekey time 108
```

(Optional) Configures a time-based rekey or a volume-based rekey for SSH.

Step 5 **exit**

Example:

```
Device(config)# exit
```

Exits global configuration mode and returns to privileged EXEC mode.

Step 6 **show ip ssh**

Example:

```
Device# show ip ssh
```

(Optional) Verifies that the SSH server is enabled and displays the version and configuration data for the SSH connection.

Invoke the SSH Client

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

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

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

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

Step 2 **ssh** **-l** *username* **-vrf** *vrf-name ip-address*

Example:

```
Device# ssh -l user1 -vrf vrf1 192.0.2.1
```

Invokes the SSH client to connect to an IP host or address in the specified virtual routing and forwarding (VRF) instance.

Configuration examples for Secure Shell

Learn how to configure Secure Shell (SSH) on your device using the following procedure and configuration examples.

Example: Configure an SSH Server

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

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

Procedure

Configure an SSH Server

```
Device> enable
Device# configure terminal
Device(config)# ip ssh timeout 30
Device(config)# end
```

Example: Invoking an SSH Client

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

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

Procedure

Invoke an SSH Client.

```
Device> enable
Device# ssh -l user1 -vrf vrf1 192.0.2.1
```

Verify SSH

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

Procedure

Step 1 Use the **show ip ssh** command to verify that SSH is enabled and to display version and configuration data.

Example:

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

The above output shows that SSH is enabled on the device.

Step 2 Use the **show ip ssh** command to check if SSH is disabled.

Example:

```
Device# show ip ssh
            %SSH has not been enabled
```

The above output indicates that SSH is not enabled on the device.

Step 3 Use the **show ssh** command to verify the status of SSH server connections.

Example:

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

This output shows the SSH server connections on the device when SSH is enabled.

Step 4 Use the **show ssh** command to check if SSH server connections are running.

Example:

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

This output indicates that there are no SSH server connections running on the device.