

Secure Shell

This section contains the following topics:

- Secure shell, on page 1
- How to configure secure shell, on page 3
- Secure copy protocol, on page 8
- SSH rekeying, on page 10
- Additional references, on page 15

Secure shell

Secure Shell (SSH) is a protocol that

- provides a secure, remote connection to a device
- provides more security for remote connections than Telnet by providing strong encryption when a device is authenticated, and
- supports SSH Version 1 (SSHv1) and SSH Version 2 (SSHv2) in this software release.

Prerequisites for configure secure shell

The following are the prerequisites for configuring the device for secure shell (SSH):

- For SSH to work, the switch needs an RSA public/private key pair.
- 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. Use the **hostname** and **ip domain-name** commands in global configuration mode.

Restrictions for configuring secure shell

Consider these restrictions when configuring the router for secure shell:

- The router supports RSA authentication.

- SSH supports only the execution-shell application.
- The SSH server and the SSH client are supported only on Data Encryption Standard (DES) (56-bit) and 3DES (168-bit) data encryption software. In DES software images, DES is the only encryption algorithm available. In 3DES software images, both DES and 3DES encryption algorithms are available.

Note Cisco highly recommends the 3DES encryption as it is stronger. See the Cisco IOS-XE Device hardening guide at https://www.cisco.com/c/en/us/support/docs/ip/access-lists/13608-21.html for details.

- This software release supports IP Security (IPSec).
- The router supports the Advanced Encryption Standard (AES) encryption algorithm with a 128-bit key, 192-bit key, or 256-bit key. However, symmetric cipher AES to encrypt the keys is not supported.
- The login banner is not supported in Secure Shell Version 1. It is supported in Secure Shell Version 2, which Cisco recommends due to its better security.
- The -l keyword and userid :{number} {IP-address} delimiter and arguments are mandatory when configuring the alternative method of Reverse SSH for console access.

SSH and router access

Secure Shell (SSH) is a protocol that

- provides a secure, remote connection to a device
- provides more security for remote connections than Telnet does by providing strong encryption when a device is authenticated, and
- supports SSH Version 1 (SSHv1) and SSH Version 2 (SSHv2).

SSH protocol characteristics

SSH functions the same in IPv6 as in IPv4. For IPv6, SSH supports IPv6 addresses and enables secure, encrypted connections with remote IPv6 nodes over an IPv6 transport.

SSH servers, integrated clients, and supported versions

The Secure Shell (SSH) Integrated Client feature is an application that

- runs over the SSH protocol to provide device authentication and encryption
- enables a Cisco device to make a secure, encrypted connection to another Cisco device or to any other device running the SSH server, and
- allows for secure communication over an unsecured network with authentication and encryption.

SSH server and client compatibility

The SSH server and SSH integrated client are applications that run on the switch. This connection provides functionality similar to that of an outbound Telnet connection except that the connection is encrypted.

SSH compatibility includes these components:

- SSH server works with the SSH client supported in this release and with non-Cisco SSH clients
- SSH client works with publicly and commercially available SSH servers
- SSH client supports the ciphers of Data Encryption Standard (DES), 3DES, and password authentication



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

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

- TACACS+
- RADIUS
- Local authentication and authorization

Recommendation: SSH configuration guidelines

Follow these guidelines when configuring the device as an SSH server or SSH client:

- An RSA key pair generated by a SSHv1 server can be used by an SSHv2 server, and the reverse.
- If you get CLI error messages after entering the **crypto key generate RSA global** configuration command, an RSA key pair has not been generated. Reconfigure the hostname and domain, and then enter the **crypto key generate RSA** command.
- When generating the RSA key pair, the message *No hostname specified* might appear. If it does, you must configure an IP hostname by using the **hostname** global configuration command.
- When generating the RSA key pair, the message No domain specified might appear. If it does, you must configure an IP domain name by using the **IP domain-name** global configuration command.
- When configuring the local authentication and authorization authentication method, make sure that AAA is disabled on the console.

How to configure secure shell

This reference provides the configuration parameters and settings required to establish secure remote connections to network devices using Secure Shell (SSH).

Set up the router to run SSH

SSH provides secure remote access to network devices. This configuration enables encrypted communication and helps protect against unauthorized access attempts.



Note Ensure that you configure user authentication for local or remote access.

Follow these steps to set up the router to run SSH.

Procedure

Step 1 Use the **enable** command to enter privileged EXEC mode.

Example:

```
Router> enable
```

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Router# configure terminal
```

Step 3 Use the **hostname** *your_hostname* and **ip domain-name** *your_domain_name* commands to configure a hostname and domain name for the device.

Example:

```
Router(config)# hostname <your_hostname>
Router(config)# ip domain-name <your_domain_name>
```

Step 4 Use the **crypto key generate rsa** command to generate an RSA key pair to enable SSH.

Example:

```
Router(config)# crypto key generate rsa
```

Note
Use a modulus size of at least 1024 bits for security.

Step 5 Use the **end** command to exit global configuration mode.

Example:

```
Router(config)# end
```

Step 6 Use the **show running-config** command to verify the configuration.

Example:

```
Router# show running-config
```

Step 7 (Optional) Use the **copy running-config startup-config** command to save the configuration.

Example:

```
Router# copy running-config startup-config
```

The router is now configured to accept SSH connections. You can connect remotely using an SSH client with the configured hostname and the appropriate user credentials.

Configure the SSH server

Configure the SSH server on a device to enable secure remote CLI access and to control SSH protocol parameters. Use this procedure when you need to set up the device as an SSH server to provide secure command-line interface access for remote management.

Note This procedure is required only if you are configuring the device as an SSH server.

Follow these steps to configure the SSH server.

Procedure

Step 1 Use the **enable** command to enable privileged EXEC mode. Enter your password if prompted.

Example:

```
Router> enable
```

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Router# configure terminal
```

Step 3 (Optional) Use the **ip ssh version** [**2**] command to configure the device to run SSH Version 2.

Example:

```
Router(config)# ip ssh version 2
```

Note
If you do not enter this command or do not specify a keyword, the SSH server selects the latest SSH version supported by the SSH client. For example, if the SSH client supports SSHv1 and SSHv2, the SSH server selects SSHv2.

Step 4 Use the **ip ssh** {**timeout** *seconds* | **authentication-retries** *number*} command to configure the SSH control parameters.

Example:

```
Router(config)# ip ssh timeout 90
Router(config)# ip ssh authentication-retries 2
```

Note

- Specify the timeout value in seconds; the default is 120 seconds, and the range is 0 to 120 seconds. This parameter applies to the SSH negotiation phase. After the connection is established, the device uses the default timeout values of the CLI-based sessions.
- By default, up to five simultaneous, encrypted SSH connections for multiple CLI-based sessions over the network are available (session 0 to session 4). After the execution shell starts, the CLI-based session timeout value returns to the default of 10 minutes.

- Specify the number of times that a client can re-authenticate to the server. The default is 3, and the range is 0 to 5.
- Repeat this step when configuring both parameters.

Step 5 (Optional) Use the **line vty** *line-number* [*ending-line-number*] command to enter line configuration mode and configure the virtual terminal line settings. Then, use the **transport input ssh** command to limit the device to SSH connections only.

Example:

```
Router(config)# line vty 1 10
Router(config-line)# transport input ssh
```

Note

- For the *line-number* and *ending-line-number* arguments, the range is from 0 to 15.
- The `transport input ssh` command specifies that the device prevents non-SSH Telnet connections, limiting the device to only SSH connections.

Step 6 Use the **end** command to exit line configuration mode and return to privileged EXEC mode.

Example:

```
Router(config-line)# end
```

Step 7 Use the **show running-config** command to verify your entries.

Example:

```
Router# show running-config
```

Step 8 (Optional) Use the **copy running-config startup-config** command to save your entries in the configuration file.

Example:

```
Router# copy running-config startup-config
```

The SSH server is configured and ready to accept secure connections with the specified parameters for version, timeout, and authentication settings.

Monitor the SSH configuration and status

This reference provides commands to monitor and display SSH server configuration and status information.

Table 1: Commands for displaying the SSH server configuration and status

Command	Purpose
show ip SSH	Shows the version and configuration information for the SSH server.
show SSH	Shows the status of the SSH server.

Configure the router for local authentication and authorization

You can configure AAA to operate without a server by setting the router to implement AAA in local mode. The router then handles authentication and authorization. No accounting is available in this configuration.



Note To secure the router for HTTP access by using AAA methods, you must configure the router with the `ip http authentication aaa` global configuration command. Configuring AAA authentication alone does not secure the router for HTTP access.

Follow these steps to configure AAA to operate without a server by setting the router to implement AAA in local mode.

Procedure

Step 1 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Router# configure terminal
```

Step 2 Use the **aaa new-model** command to enable AAA.

Example:

```
Router(config)# aaa new-model
```

Step 3 Use the **aaa authentication login default local** command to set the login authentication to use the local username database.

Example:

```
Router(config)# aaa authentication login default local
```

Note
The `default` keyword applies the local user database authentication to all ports.

Step 4 Use the **aaa authorization exec local** command to configure user AAA authorization, check the local database, and allow the user to run an EXEC shell.

Example:

```
Router(config)# aaa authorization exec local
```

Step 5 Use the **aaa authorization network local** command to configure user AAA authorization for all network-related service requests.

Example:

```
Router(config)# aaa authorization network local
```

Step 6 Use the **username** *name* [**privilege** *level*] **password** *encryption-type password* command to enter the local database and establish a username-based authentication system.

Example:

```
Router(config)# username your_user_name privilege 1 password 7 secret567
```

Note

- Repeat this command for each user.
- For *name*, specify the user ID as one word. Spaces and quotation marks are not allowed.
- (Optional) For *level*, specify the privilege level the user has after gaining access. The range is 0 to 15. Level 15 gives privileged EXEC mode access; level 0 gives user EXEC mode access.
- For *encryption-type*, enter 0 to specify that an unencrypted password follows, or enter 7 to specify that a hidden password follows.
- For *password*, specify the password the user must enter to gain access to the router. The password must be from 1 to 25 characters, can contain embedded spaces, and must be the last option specified in the `username` command.

Step 7 Use the **end** command to exit configuration mode and return to privileged EXEC mode.

Example:

```
Router(config)# end
```

Step 8 Use the **show running-config** command to verify your entries.

Example:

```
Router# show running-config
```

Step 9 (Optional) Use the **copy running-config startup-config** command to save your entries in the configuration file.

Example:

```
Router# copy running-config startup-config
```

The router is configured for local authentication and authorization using AAA methods. Users can authenticate using the local username database, and authorization is handled locally by the router.

Secure copy protocol

The Secure Copy Protocol (SCP) is a network protocol that

- provides a secure and authenticated method for copying router configuration or router image files
- relies on Secure Shell (SSH), an application and a protocol that provide a secure replacement for the Berkeley r-tools, and
- enables secure file transfer operations between network devices.

Requirements for secure copy configuration

Before enabling SCP, you must correctly configure SSH, authentication, and authorization on the switch.

- Because SCP relies on SSH for its secure transport, the router must have an RSA key pair.
- SCP relies on SSH for security.

- SCP requires that authentication, authorization, and accounting (AAA) authorization be configured so the router can determine whether the user has the correct privilege level.
- A user must have appropriate authorization to use SCP.
- A user who has appropriate authorization can use SCP to copy any file in the Cisco IOS File System (IFS) to and from a switch by using the **copy** command. An authorized administrator can also do this from a workstation.

Restrictions for configuring secure copy

Before enabling SCP, you must correctly configure SSH, authentication, and authorization on the router.

When using SCP, you cannot enter the password into the **copy** command. You must enter the password when prompted.

Configure secure copy

This task enables SCP server functionality on a Cisco router to allow secure file transfers between the router and remote hosts. SCP provides a secure method for copying files to and from network devices.



Note Before you configure SCP, ensure that proper authentication mechanisms are in place.

Follow these steps to configure the Cisco router for Secure Copy (SCP) server-side functionality.

Procedure

Step 1 Use the **enable** command to enable privileged EXEC mode. Enter your password if prompted.

Example:

```
Router> enable
```

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Router# configure terminal
```

Step 3 Use the **aaa new-model** command to enable the AAA access control model.

Example:

```
Router(config)# aaa new-model
```

Step 4 Use the **aaa authentication login** {**default** | *list-name*} *method1* [*method2...*] command to set AAA authentication at login.

Example:

```
Router(config)# aaa authentication login default group tacacs+
```

Step 5 Use the **username** *name* [**privilege** *level*] **password** *encryption-type encrypted-password* command to establish a username-based authentication system.

Example:

```
Router(config)# username superuser privilege 2 password 0 superpassword
```

Note
You can omit this step if a network-based authentication mechanism, such as TACACS+ or RADIUS, is configured.

Step 6 Use the **ip scp server enable** command to enable SCP server-side functionality.

Example:

```
Router(config)# ip scp server enable
```

Step 7 Use the **exit** command to exit global configuration mode and return to privileged EXEC mode.

Example:

```
Router(config)# exit
```

Step 8 (Optional) Use the **show running-config** command to display the SCP server-side functionality.

Example:

```
Router# show running-config
```

Step 9 (Optional) Use the **debug ip scp** command to troubleshoot SCP authentication problems.

Example:

```
Router# debug ip scp
```

SCP server functionality is now enabled on the router, allowing secure file transfers. You can now use SCP commands to copy files to and from the router.

```
Router# copy scp <somefile> your_username@remotehost:/<some/remote/directory>
```

SSH rekeying

SSH rekeying is a security feature that:

- replaces the encryption and integrity keys during an active SSH session
- initiates a new key exchange based on a configured time or data threshold, and
- helps reduce the risk of long-term key exposure.

SSH rekeying characteristics

SSH rekeying can be initiated by either the SSH client or server. During rekeying, data transfer halts until the key exchange completes. The encryption algorithm negotiated at session startup remains in use during rekeying. OpenSSH-based clients version 7.5 or later support SSH rekey exchange.

SSH rekeying scenarios

- Rekeying every hour to ensure session security over long connections.
- Rekeying after 1 GB of data transfer to limit the data encrypted with a single key.

SSH rekeying types

SSH rekeying types are configuration methods that:

- enable periodic renewal of encryption keys during SSH sessions
- provide time-based and volume-based triggering options using the **ip SSH rekey** command, and
- enhance security by preventing prolonged use of the same encryption keys.

Configuration methods

You can configure SSH rekeying on Cisco routers in two ways by using the **ip SSH rekey** command:

- Time-based rekeying: Specify the interval (in minutes or seconds) after which rekeying occurs.
- Volume-based rekeying: Specify the data volume (in KB or MB) that triggers rekeying.

By default the **ip SSH rekey** command is disabled. If you configure these settings using the **ip SSH rekey** command without specifying time or volume, then it will apply both time and volume rekey with default values. The default value for time is 60 minutes and volume is 1 GB.

Rekeying by time initiates a key exchange after a specific time interval:

- In Common Criteria mode, the maximum allowed time is less than 60 minutes or 3600 seconds.
- The configurable range is 10 to 1440 minutes.
- For some devices, the default is 60 minutes, and the command accepts values in minutes.

Rekeying by volume enforces a new key exchange when the transmitted or received data reaches a configured threshold:

- In Common Criteria mode, the maximum rekey volume is 1024 MB (1 GB).
- The configurable range for the rekey volume is 100 KB to 4 GB.
- For some devices, the default is 1 GB, and the command accepts values in kilobytes.

Prerequisites for configuring SSH rekeying

To configure SSH rekeying on Cisco routers, ensure these prerequisites are met.

- The device runs Cisco IOS with SSH enabled.
- SSH version 2 is recommended.
- Administrative privilege access is required.

Best practices for SSH rekeying configuration

Best practices for SSH rekeying configuration are:

- Enable SSH rekeying to reduce the risk from potential key compromise, especially in high-security environments.
- Ensure rekey intervals comply with security policies, such as Common Criteria, by setting appropriate time or volume thresholds.

SSH session idle timeout settings

SSH sessions can be configured to disconnect after a period of inactivity.

- SSH idle timeout: **line vty <0-240>** (in minutes; 0 means never timeout)
- **exec-timeout <0-240>** (in minutes; 0 means never timeout)

Restrictions for SSH rekeying on Cisco routers

The following limitations apply to SSH rekeying on Cisco routers:

- During SCP downloads, the device acting as receiver cannot control the sender's data volume, so rekey is triggered only after the configured volume is received.
- Large SSH bulk mode settings (for example, 1 GB) can lead to fewer rekeys than expected due to larger data windows.
- SSH sessions established without rekey configuration do not support rekeying
- When rekey is enabled, existing sessions will only adopt rekeying after the next exchange initiated from the other side.
- Removing the rekey configuration disables SSH rekey for existing sessions.
- Changing the rekey configuration does not affect active sessions until the subsequent rekey exchange.
- When a rekey exchange occurs, the related data and time parameters for the session reset to the new configured value.
- Do not modify the encryption method during rekeying.

The actual number of rekeys during SCP downloads may not match calculated expectations.

This behavior is by design and aligns with current SSH implementation.

Example observation:

Transferring a 631.3 MB file with a 100 KB rekey threshold could yield an expected 6,463 rekeys, but only 689 may occur.

This discrepancy is documented and not considered a defect.

Configure SSH rekeying

SSH rekeying enhances session security by periodically renewing encryption keys. Rekeying can be based on time intervals or data volumes. The configuration involves enabling SSH, setting the SSH version, and defining the rekey parameters.

Follow these steps to configure SSH rekeying.

Procedure

Step 1 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Router# configure terminal
```

Step 2 Use the **ip domain-name** *domain-name* command to set the domain name.

Example:

```
Router(config)# ip domain-name example.com
```

Step 3 Use the **crypto key generate rsa modulus** *modulus-size* command to generate the RSA keys.

Example:

```
Router(config)# crypto key generate rsa modulus 2048
```

Step 4 Use the **ip ssh version 2** command to set the SSH version to 2.

Example:

```
Router(config)# ip ssh version 2
```

Step 5 Use the **ip ssh rekey time** *minutes* command to configure time-based rekey.

Example:

```
Router(config)# ip ssh rekey time 120
```

Note
The valid range is 10 to 1440 minutes (the default is 60 minutes).

Step 6 Use the **ip ssh rekey volume** *size-in-kb* command to configure volume-based rekey.

Example:

```
Router(config)# ip ssh rekey volume 10240
```

Note

- This example triggers a rekey after 10,240 KB.
- The valid range is 100 to 4194303 KB (the default is 1 GB).

Step 7 (Optional) Use the **no ip ssh rekey time** or **no ip ssh rekey volume** command to disable time-based or volume-based rekeying.

Example:

```
Router(config)# no ip ssh rekey time
```

Example:

```
Router(config)# no ip ssh rekey volume
```

SSH rekeying is enabled on the device, improving session security through periodic key renewal.

Verify SSH rekey configuration

Verification ensures that SSH rekeying is correctly configured and operational on the device.

Follow these steps to verify the SSH rekey configuration.

Procedure

Step 1 Use the **show ip ssh** command to display the SSH configuration details.

Example:

```
Router# show ip ssh
            SSH Enabled - version 2.0
            Authentication methods:publickey,keyboard-interactive,password
            Authentication Publickey
Algorithms:ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,ssh-ed25519,x509v3-ecdsa-sha2-nistp256,x509v3-ecdsa-sha2-nistp384,x509v3-ecdsa-sha2-nistp521,rsa-sha2-256,rsa-sha2-512,x509v3-rsa2048-sha256

            Hostkey
Algorithms:ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,rsa-sha2-512,rsa-sha2-256

            Encryption
Algorithms:chacha20-poly1305@openssh.com,aes128-gcm@openssh.com,aes256-gcm@openssh.com,aes128-gcm,aes256-gcm,aes128-ctr,aes192-ctr,aes256-ctr

            MAC
Algorithms:hmac-sha2-256-etm@openssh.com,hmac-sha2-512-etm@openssh.com,hmac-sha2-256,hmac-sha2-512

            KEX
Algorithms:curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-group14-sha256,diffie-hellman-group16-sha512

            Authentication timeout: 120 secs; Authentication retries: 3
            Minimum expected Diffie Hellman key size : 2048 bits
            IOS Keys in SECSH format(ssh-rsa, base64 encoded): TP-self-signed-1538213291
            Modulus Size : 2048 bits
           ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCw/l05OHVk/3MAC30fIpLf6rphTETPZtnIdlYbZPCc

           L+gLy+dxsxq8kYXZ5lTQq+lstB+PB868xDz/mkk9OfNFcEwa2SavoyHuDeqN58FMFmU4z9guwfe66dIk

           sdt1qA086Wc45BUBYrE3R4uVC1gQdkuHyIZmN54c3lvSRuToaGNo3M10rRjIELOWOV1Qbcr8e0nrAAmf

           Xznx2lDqO2QPdynLEpLWinLpnwEFEJtQ2DXRAZO+a1SqGvfTbLZ+EQ7SCI39X0xC5YVCFRjD3NF+6NpO

            Q6wt0my/5xNXr1m5uabfOco96lAXbCD4w55JDuv0Eq3siefVAuXXtgGpDCVB
            IOS Keys in SECSH format(ssh-ec, base64 encoded): NONE
```

Note

The SSH session attributes and algorithms include:

- SSH version and enablement status
- Supported authentication methods and algorithms (public key, keyboard-interactive, password)
- Hostkey and encryption algorithms

- MAC and key exchange algorithms
- Authentication timeout and retry settings
- Minimum expected Diffie-Hellman key size
- Current session keys (in SECSH format)

Step 2 Use the **show running-config | include ssh** command to display the SSH-related running configuration details.

Example:

```
Router# show running-config | include ssh
            ip ssh bulk-mode 13107245
            ip ssh rekey volume 100
            transport input ssh
            transport input ssh
```

Example:

```
Router# show running-config | include ssh
            ip ssh bulk-mode 13107245
            ip ssh rekey time 120
            transport input ssh
            transport input ssh
```

Step 3 Use the **show ssh rekey** command to monitor SSH rekey activity and current session information.

Example:

```
Router# show ssh rekey
            SSHv2 server sessions
            Connection  TimeTillRekey  DataLeftTillRekey TimeRekeys  VolumeRekeys
            0           0:00           98                 0           0

            %No SSHv2 client connections running.
```

Example:

```
Router# show ssh rekey
            SSHv2 server sessions
            Connection  TimeTillRekey  DataLeftTillRekey TimeRekeys  VolumeRekeys
            0           0:00           99997              0           0

            SSHv2 client sessions
            Connection  TimeTillRekey  DataLeftTillRekey TimeRekeys  VolumeRekeys
            1           0:00           0                  0           0
```

Additional references

This sections provide references related to the SSH feature.

Related Topic	Document Title
Configuring Identity Control policies and Identity Service templates for Session Aware networking.	Session Aware Networking Configuration Guide, Cisco IOS XE Release 3SE: https://www.cisco.com/en/US/docs/ios-xml/ios/san/configuration/xe-3se/3850/san-xe-3se-3850-book.pdf

Related Topic	Document Title
Configuring RADIUS, TACACS+, Secure Shell, 802.1X and AAA.	Secure Shell Configuration Guide, Cisco IOS XE Gibraltar 16.11.x: https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst9500/software/release/16-11/configuration_guide/sec/b_1611_sec_9500_cg/configuring_secure_shell__ssh_.html