Contents

1 Configuring SSH

1.1 Introduction

1.1.1 Overview

Secure Shell (SSH) is a network security protocol that requires encryption and authentication and is used for remote access and file transfer.

SSH is based on the server/client structure. The device supports both the SSH server and client functions. As an SSH server, the device can connect to multiple SSH clients. As an SSH client, the device allows users to establish SSH connections with devices that support the SSH server function.

The SSH function is similar to the Telnet service. However, the encryption and authentication features of SSH can ensure stronger security guarantee for users. When users log in to the device through an insecure network environment, SSH can effectively protect the device against attacks, such as IP address spoofing and plaintext password interception.

1.1.2 Principles

1. SSH Version

SSHv1 and SSHv2 are available, and the two are mutually exclusive. Compared with SSHv1, SSHv2 provides higher performance and security.

The device supports both versions and supports SSH interaction using the IPv4 and IPv6 addresses. Unless otherwise specified, SSH in this document refers to SSHv2.

2. SSH Exchange Process

Interaction between an SSH client and an SSH server is classified into seven phases: connection establishment, version negotiation, key exchange and algorithm negotiation, user authentication, session request, session interaction, and session closure.

- Connection establishment

a The server listens on Transmission Control Protocol (TCP) port 22 and waits for a connection request from the client.

b The client initiates a TCP connection request to port 22 of the server to establish a TCP connection with the server.

- Version negotiation

  c The server sends a version negotiation packet to the client.

d The client receives and processes the packet and replies the version to be used to the server.

e The server processes the reply from the client and determines whether version negotiation is successful.

- Key exchange and algorithm negotiation

  If the version negotiation succeeds, key exchange and algorithm negotiation are performed.

  f The server and the client exchange the algorithm negotiation packet with each other and determine the final algorithm based on their capacity.

gThe server and the client work together to generate a session key and session ID according to the key exchange algorithm and host key, which will be applied to subsequent user authentication, data encryption, and data decryption.

- User authentication

  After the encrypted channel is set up, user authentication is performed.

  hThe client sends an authentication request to the server.

iThe server repeatedly conducts authentication for the client until the authentication succeeds or the server shuts down the connection because the maximum number of authentication attempts is reached.

- Session request

  jAfter the authentication is successful, the client sends a session request to the server.

  kThe server waits and processes the client's session request. After the session request is successfully processed, the SSH client and server enter the session interaction phase.

- Session interaction

  In this phase, bidirectional transmission and processing of encrypted data are allowed.

  lThe client sends commands that need to be executed to the server.

  mThe server decrypts, parses, and processes these commands.

nThe server encrypts and sends the processing result to the client.

oThe client decrypts and processes the message from the server.

- Session closure

  The server and client disconnect the connection, and close the session.

3. User Authentication Mechanism

SSH supports password authentication, public key authentication, and other client authentication mechanisms.

- Password authentication

  Password authentication authenticates a client by verifying the username and password.

aThe client sends encrypted username and password to the server to request password authentication.

bAfter receiving the request, the server decrypts the information, authenticates the client's identity through the Authentication, Authorization and Accounting (AAA) service (including local authentication and remote authentication), and sends an authentication success or failure message to the client.

- Public key authentication

  Public key authentication uses digital signature algorithms, such as Rivest-Shamir-Adleman (RSA) and Digital Signature Algorithm (DSA) to authenticate a client.

  cThe client sends a public key authentication request to the server. This request contains information, including the username, public key, and public key algorithm.

dAfter receiving the request, the server checks whether the public key is valid.

eIf not, the server directly returns an authentication failure message. If yes, the server performs digital signature authentication on the client and returns a message indicating successful or failed authentication.

Instruction

Public key authentication applies only to SSHv2 clients.

4. Public Key Algorithm

A public key algorithm is an asymmetric encryption algorithm used to encrypt communication data between an SSH client and server to ensure data transfer security.

The device supports three public key algorithms: RSA, DSA, and Elliptic Curves Cryptography (ECC).

- RSA

  RSA is based on most factorization algorithms and can be used for key and signature exchange. Due to low encryption and decryption efficiency, it does not apply to encryption and decryption of a large number of data.

- DSA

  Compared with RSA, DSA can be used only for signature and cannot be used for data encryption and decryption or key exchange. However, its efficiency is much higher than that of RSA.

- ECC

  ECC is based on the discrete logarithm algorithm and is difficult to be cracked down. In same security conditions, ECC has higher processing efficiency and lower storage and bandwidth requirements than RSA and DSA.

1.1.3 Protocols and Standards

- RFC 4251: The Secure Shell (SSH) Protocol Architecture
- RFC 4252: The Secure Shell (SSH) Authentication Protocol
- RFC 4253: The Secure Shell (SSH) Transport Layer Protocol
- RFC 4254: The Secure Shell (SSH) Connection Protocol
- RFC 4419: Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol
- RFC 4716: The Secure Shell (SSH) Public Key File Format
- RFC 4819: Secure Shell Public Key Subsystem
- RFC 3526: More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)
- RFC 2409: The Internet Key Exchange (IKE)
- RFC 1950: ZLIB Compressed Data Format Specification version 3.3

1.2 Configuration Task Summary

All the configuration tasks below are optional. Select the configuration tasks as required.

(1)Configuring SSH Server

- Enabling SSH Server
- (Optional) Configuring the SSH Server Version
- (Optional) Configuring the Listening Port

- (Optional) Configuring the Key and Key Exchange Algorithm of the SSH Server
- (Optional) Configuring the Encryption Algorithm
- (Optional) Configuring Authentication Parameters
- (Optional) Configuring an ACL
- (Optional) Configuring IP Address Blocking
- (Optional) Disconnecting an SSH Client Connection

(2)Configuring SCP Server

(3)Configuring SSH Client

- Establishing a Connection with the SSH Server
- (Optional) Restoring an Established SSH Client Session
- (Optional) Disconnecting a Suspended SSH Client Session

(4)Configuring SCP Client

1.3 Configuring SSH Server

1.3.1 Overview

After the SSH server function is configured on a device, SSH clients can remotely log in to the device.

1.3.2 Configuration Tasks

SSH server configuration includes the following tasks:

- Enabling SSH Server
- (Optional) Configuring the SSH Server Version
- (Optional) Configuring the Listening Port
- (Optional) Configuring the Key and Key Exchange Algorithm of the SSH Server
- (Optional) Configuring the Encryption Algorithm
- (Optional) Configuring Authentication Parameters
- (Optional) Configuring an ACL
- (Optional) Configuring IP Address Blocking
- (Optional) Disconnecting an SSH Client Connection

1.3.3 Enabling SSH Server

(1)Enter the privileged EXEC mode.

enable

(2)Enter the global configuration mode.

configure terminal

(3)Enable the SSH server function.

enable service ssh-server

The SSH server function is disabled by default.

(4)(Optional) Enable the SSHv1 function.

ip ssh compatible-ssh1x enable

The SSHv1 function is disabled by default.

To use the SSHv1 version, enable the SSHv1 function first.

1.3.4 Configuring the SSH Server Version

1. Overview

The SSH server supports the SSHv1 and SSHv2 versions. Compared with SSHv1, SSHv2 has higher performance and security. During actual configuration, select a proper SSH server version based on the client version and security requirements.

2. Restrictions and Guidelines

The SSH server is compatible with SSHv1 and SSHv2 clients by default, that is, clients of both versions can connect to the SSH server. After an SSH server version is configured, only clients of the corresponding version can connect to the server.

3. Procedure

(1)Enter the privileged EXEC mode.

enable

(2)Enter the global configuration mode.

configure terminal

(3)Configure the SSH server version.

ip ssh version *version-type*

The SSH server is compatible with the SSHv1 and SSHv2 clients by default.

1.3.5 Configuring the Listening Port

1. Overview

An SSH server uses port 22 to listen to client connections by default. The listening port can be changed through configuration.

Port 22 is a protocol-defined standard port and vulnerable to attacks from malicious users. If an attacker accesses this port frequently in a short time, problems such as bandwidth waste and performance deterioration may occur. As a result, other users cannot access the server (that is, the DoS attack). Changing the listening port can effectively prevent the problems.

2. Procedure

(1)Enter the privileged EXEC mode.

enable

(2)Enter the global configuration mode.

configure terminal

(3)Configure the listening port of the SSH server.

ip ssh port *ssh-monitor-port*

The default listening port of the SSH server is port **22**.

1.3.6 Configuring the Key and Key Exchange Algorithm of the SSH Server

1. Overview

The key and key exchange algorithm of the SSH server are configured to establish a communication connection between the client and server.

2. Restrictions and Guidelines

SSHv1 uses an RSA key, and SSHv2 uses an RSA or a DSA key. If an RSA key is generated, both SSHv1 and SSHv2 can use the key. If a DSA key is generated, only SSHv2 can use the key.

3. Procedure

(1)Enter the privileged EXEC mode.

enable

(2)Enter the global configuration mode.

configure terminal

(3)Generate the public key of the SSH server.

crypto key generate { **dsa** | **ecc** | **rsa** }

No public key is generated on the SSH server by default.

If no key is generated on the SSH server, this parameter must be configured. To delete the public key of the SSH server, run the **crypto key zeroize** { **dsa** | **ecc** | **rsa** } command instead of the **no** form of this command.

(4)Configure the DH key exchange algorithms supported by the SSH server.

ip ssh key-exchange { **dh_group_exchange_sha1** | **dh_group14_sha1** | **dh_group1_sha1** | **ecdh_sha2_nistp256** | **ecdh_sha2_nistp384** | **ecdh_sha2_nistp521** }

Orion SSHv2 servers support diffie-hellman-group-exchange-sha1, diffie-hellman-group14-sha1, ecdh_sha2_nistp256, ecdh_sha2_nistp384, and ecdh_sha2_nistp521 for key exchange, and SSHv1 servers support none by default.

1.3.7 Configuring the Encryption Algorithm

(1)Enter the privileged EXEC mode.

enable

(2)Enter the global configuration mode.

configure terminal

(3)Configure the encryption modes supported by the SSH server.

ip ssh cipher-mode { **cbc** | **ctr** | **gcm** | **others** } *

The encryption modes supported by the SSH server are **ctr** and **gcm** by default.

It is approved that **cbc** and **others** encryption algorithms can be decrypted in a limited period of time. Therefore, organizations or companies that have high security requirements can set the encryption modes supported by the SSH server to **ctr** and **gcm** to enhance the security level of the SSH server.

(4)Configure the message authentication algorithms supported by the SSH server.

ip ssh hmac-algorithm { **md5** | **md5-96** | **sha1** | **sha1-96** | **sha2-256** | **sha2-512** }

SSHv1 servers do not support any message authentication algorithms, and SSHv2 servers support the MD5, SHA1, SHA1-96, MD5-96, sha2-256, and sha2-512 message authentication algorithms by default.

1.3.8 Configuring Authentication Parameters

1. Overview

Parameters related to user authentication include the authentication timeout time, number of authentication attempts, and public key authentication.

2. Procedure

(1)Enter the privileged EXEC mode.

enable

(2)Enter the global configuration mode.

configure terminal

(3)Configure the user authentication timeout time on the SSH server.

ip ssh time-out *timeout-time*

The default user authentication timeout time on the SSH server is **120** seconds.

If authentication still does not succeed when the user authentication timeout time expires, user authentication fails.

(4)Configure the number of authentication attempts allowed on the SSH server.

ip ssh authentication-retries *retry-times*

The default maximum number of authentication attempts allowed on the SSH server is **3**.

If the number of authentication attempts of a user by using the account and password exceeds the configured number of authentication attempts, user authentication fails.

(5)Associate with the public key file and username of a client.

ip ssh peer *username* **public-key** { **dsa** | **ecc** | **rsa** } *filename-path*

If a client needs to use public key authentication, the client's public key and username need to be associated with the device. Only SSHv2 supports public key authentication.

1.3.9 Configuring an ACL

1. Overview

An ACL can filter out users who meet specific rules. This prevents unauthorized users from accessing the device through SSH and causing security risks.

2. Procedure

(1)Enter the privileged EXEC mode.

enable

(2)Enter the global configuration mode.

configure terminal

(3)Configure an ACL on the SSH server.

(IPv4 address)

ip ssh access-class { *acl-name* | *acl-number* }

No IPv4 ACL is configured on the SSH server by default.

(IPv6 address)

ipv6 ssh access-class *acl-name*

No IPv6 ACL is configured on the SSH server by default.

1.3.10 Configuring IP Address Blocking

1. Overview

When the number of authentication failures for login through SSH reaches the configured limit in an authentication failure count period, the source IP address is blocked. That is, the SSH client that uses this source IP address is not allowed to log in to the device to prevent the device being attacked. The SSH client can log in to the device only after the IP address is awakened. This function is enabled by default and can be manually disabled.

2. Procedure

(1)Enter the privileged EXEC mode.

enable

(2)Enter the global configuration mode.

configure terminal

(3)Enable IP address blocking on the SSH server.

ip ssh ip-block disable

The IP address blocking function of the SSH server is enabled by default.

Run the **no** form of this command to disable this feature.

(4)(Optional) Configure the number of authentication failures for blocking an IP address and the time period for counting authentication failures on the SSH server.

ip ssh ip-block failed-times *failed-times* **period** *period-time*

The allowed maximum number of authentication failures is **6**, and the time period for counting authentication failures is **5** minutes by default.

(5)(Optional) Configure the time period for awakening blocked IP addresses on the SSH server.

ip ssh ip-block reactive *reactive-interval*

Blocked IP addresses are awakened every **5** minutes by default.

After the time period for awakening blocked IP addresses is reached, SSH clients using these IP addresses can log in to the SSH server again.

(6)(Optional) Clear user information with blocked IP addresses and authentication failures.

clear ssh ip-block { **all** | *ipv4-address* | *ipv6-address* }

After the user information is cleared, corresponding clients can log in to the SSH server again.

1.3.11 Disconnecting an SSH Client Connection

(1)Enter the privileged EXEC mode.

enable

(2)Disconnect an established SSH client connection.

disconnect ssh { **vty** *session-id* | *session-id* }

Serving as an SSH server, the device may connect to multiple SSH clients. You can run this command to forcibly disconnect a client from the device. The client disconnection methods are as follows:

- Specify an SSH session ID. To display the SSH session ID of a client, run the **show ssh** command.
- Specify a Virtual Teletype (VTY) session ID. To display the VTY session ID of a client, run the **show users** command. This command can be used to disconnect SSH connections only.

1.4 Configuring SCP Server

1.4.1 Overview

Secure copy protocol (SCP) is a protocol that supports network file transfer. SCP is implemented based on RCP. RCP is responsible for file transfer, and SSH provides the authentication and encryption functions for RCP.

After the SCP server function is configured on the device, users can run the **scp** command to upload files to or download files from the device. Data exchanged during the process is encrypted for security.

1.4.2 Restrictions and Guidelines

The SSH server function must be enabled before the SCP server function is configured.

1.4.3 Procedure

(1)Enter the privileged EXEC mode.

enable

(2)Enter the global configuration mode.

configure terminal

(3)Enable the SCP server function.

ip scp server enable

The SCP server function is disabled by default.

(4)(Optional) Configure the transmission path for uploading files to or downloading files from the SCP server.

ip scp server topdir { **flash:/***path* | **tmp:/***path* | **usb0:/***path* }

The default transmission path for file upload and download is **flash:/**.

1.5 Configuring SSH Client

1.5.1 Overview

When the local device is configured as an SSH client, the device can establish an SSH connection to another device that provides the SSH server service to remotely log in to the device for management operations.

1.5.2 Configuration Tasks

SSH client configuration includes the following tasks:

(1)Establishing a Connection with the SSH Server

(2)(Optional) Restoring an Established SSH Client Session

(3)(Optional) Disconnecting a Suspended SSH Client Session

1.5.3 Establishing a Connection with the SSH Server

(1)Enter the privileged EXEC mode.

enable

(2)Establish a remote encrypted session with the SSH server as an SSH client.

ssh [**oob**] [**-v** { **1** | **2** } | **-c** { **3des** | **aes128-cbc** | **aes192-cbc** | **aes256-cbc** | **aes128-ctr** | **aes192-ctr** | **aes256-ctr** | **aes128-gcm** | **aes256-gcm** } | **-l** *username* | **-m** { **hmac-md5-96** | **hmac-md5-128** | **hmac-sha1-96** | **hmac-sha1-160** | **hmac-sha2-256** | **hmac-sha2-512** } | **-p** *port-num*] * { *ip-address* | *hostname* } [**via** *mgmt-name*] [**/source** { **ip** *ipv4-address* | **ipv6** *ipv6-address* | **interface** *interface-type interface-number* }] [**/vrf** *vrf-name*]

SSHv1 supports only the DES (56-bit key) and 3DES (168-bit key) encryption algorithms. SSHv2 supports the following Advanced Encryption Standards (AES): ASE128-CBC, AES192-CBC, AES256-CBC, AES128-CTR, AES192-CTR, and AES256-CTR. If you specify an unmatched encryption or authentication algorithm when selecting an SSH version, the unmatched algorithm will be ignored when a connection is established.

(3)Enter the global configuration mode.

configure terminal

(4)(Optional) Configure the source interface of the SSH client.

ip ssh source-interface *interface-type interface-number*

No SSH client source interface is configured by default.

After the source interface is configured, the SSH client uses the IP address on this interface as the global source address during communication. If no source interface is configured, the source address of SSH packets will be obtained by querying the corresponding route based on the destination address. If no source interface or source IP address is independently specified for an SSH connection, the global configuration is used.

1.5.4 Restoring an Established SSH Client Session

1. Overview

After the device establishes an SSH session with the SSH server as a client, you can press Ctrl+Shift+6+X to exit the session temporarily. After exiting the session, you can run the corresponding command to restore the session.

2. Restrictions and Guidelines

To display information about the SSH connections established by the device as an SSH client, run the **show ssh-session** command.

3. Procedure

(1)Enter the privileged EXEC mode.

enable

(2)Restore an established SSH client session.

ssh-session *session-id*

1.5.5 Disconnecting a Suspended SSH Client Session

1. Overview

After the device connects to the SSH server as an SSH client, you can run the corresponding command to disconnect an SSH session with the specified session ID.

2. Procedure

(1)Enter the privileged EXEC mode.

enable

(2)Disconnect a suspended SSH client session.

disconnect ssh-session *session-id*

You can run the **show ssh-session** command to display *session-id* of the device.

1.6 Configuring SCP Client

1.6.1 Overview

When the local device is configured as an SCP client, the device can run the **scp** command to upload files to or download files from a device that provides the SCP server service. Data exchanged during the process is encrypted for security.

1.6.2 Procedure

(1)Enter the privileged EXEC mode.

enable

(2)Enter the global configuration mode.

configure terminal

(3)(Optional) Configure the source interface of the SCP client.

ip scp client source-interface *interface-type interface-number*

No SCP client source interface is configured by default.

After the source interface is configured, the SCP client uses the IP address on the interface as the global source address during communication. If no source interface is configured, the source address of SCP packets will be obtained by querying the corresponding route based on the destination address. If no source interface or source IP address is independently specified for an SCP connection, the global configuration is used.

(4)Upload files to or download files from the remote SCP server as an SCP client.

scp [**oob**] [**-v** { **1** | **2** } | **-c** { **3des** | **aes128-cbc** | **aes192-cbc** | **aes256-cbc** | **aes128-ctr** | **aes192-ctr** | **aes256-ctr** | **aes128-gcm** | **aes256-gcm** } | **-m** { **hmac-md5-96** | **hmac-md5-128** | **hmac-sha1-96** | **hmac-sha1-160** | **hmac-sha2-256** | **hmac-sha2-512** } | **-p** *port-num*] * *source-file destination-file* [**via** *mgmt-name*] [**/source** { **ip** *ipv4-address* | **ipv6** *ipv6-address* | **interface** *interface-type interface-number* }] [**/vrf** *vrf-name*]

1.7 Monitoring

Run the **show** commands to check the running status of a configured function to verify the configuration effect.

Run the **debug** commands to output debugging information.

Notice

System resources are occupied when debugging information is output. Therefore, disable the debugging function immediately after use.

Table 1-1 Monitoring

Command	Purpose
show ip ssh	Displays effective configurations of the SSH server.
show ssh	Displays information about established SSH connections.
show ssh ip-block { **all** | **list** }	Displays information about blocked IP addresses and authentication failures.
show crypto key mypubkey { **dsa** | **ecc** | **rsa** }	Displays partial public key information of the SSH server.
show ssh-session	Displays information about established SSH client sessions.
debug ssh	Debugs sessions of the SSH server.
debug ssh client	Debugs sessions of the SSH client.

1.8 Configuration Examples

1.8.1 Configuring Line Password Authentication for SSH Users

1. Requirements

After the SSH server function is enabled on a device, users can use the password configured in line configuration mode to log in to the device.

2. Topology

Figure 1-1 Configuring Line Password Authentication for SSH Clients

3. Notes

- Enable the SSH server function.
- Configure the line password.
- Enable password authentication in the line.

4. Procedure

(1)Enable the SSH server function.

```
Device> enable
Device# configure terminal
Device(config)# enable service ssh-server
```

(2)Set the line password to **password1**.

```
Device(config)# line vty 0 4
Device(config-line)# password password1
```

(3)Enable password authentication in the line.

```
Device(config-line)# login
```

5. Verification

- Use the client software to log in to the device through SSH. The password is **password1**, and the username can be any value. Verify that the login is successful. Use the PuTTY software as an example. The login page is as follows:

```
login as: anyname
anyname@192.168.1.1's password:



Last login: Apr  2 2021 10:00:00 from 192.168.1.2 through ssh.
Device#
```

- Run the **show ip ssh** command to verify that the SSH server takes effect.

```
Device# show ip ssh
SSH Enable - version 2.0
SSH Port:               22
SSH Cipher Mode:        ctr,gcm
SSH HMAC Algorithm:     sha2-256,sha2-512
Authentication timeout: 120 secs
Authentication retries: 3
SSH SCP Server:         disabled
SSH ip-block:           enabled
```

- Run the **show ssh** command to display the SSH connection information. Verify that a connection with username **anyname** (used during login) exists.

```
Device# show ssh
Connection Version Encryption       Hmac         Compress   State
Username
```

```
         1     2.0 aes256-ctr       hmac-sha2-256 none       Session started
anyname
```

- Run the **show users** command to display information about connected clients.

```
Device#show users
Line              User         Host(s)              Idle       Location
----------------- ------------ -------------------- ----------
------------------
   0 con 0        ---          idle                 00:00:16   ---
*  1 vty 0        ---          idle                 00:00:00   192.168.1.2
```

6. Configuration Files

```
!
enable service ssh-server
!
line console 0
line vty 0 4
 login
 password 7 $10$396$mnO2p3SVaxm7$
!
```

1.8.2 Configuring Local User Authentication for SSH Clients

1. Requirements

After the SSH server function is enabled on a device, users can use the local account created on the device to log in to the device through SSH.

2. Topology

Figure 1-1 Configuring Local User Authentication for SSH Clients

3. Notes

- Enable the SSH server function.
- Add a local user account.
- Enable local user authentication in the line.

4. Procedure

(1)Enable the SSH server function.

```
Device> enable
Device# configure terminal
Device(config)# enable service ssh-server
```

(2)Add a local user whose username is **hostname1** and password is **password1**.

```
Device(config)# username hostname1 password password1
```

(3)Enable local user authentication in the line.

```
Device(config)# line vty 0 4
Device(config-line)# login local
```

5. Verification

- Use the client software to log in to the device through SSH. The username is **hostname1**, and the password is **password1**. Verify that the login is successful. Use the PuTTY software as an example. The login page is as follows:

```
login as: hostname1
hostname1@192.168.1.1's password:



Last login: Apr  2 2021 10:00:00 from 192.168.1.2 through ssh.
Device>
```

- Run the **show ip ssh** command to verify that the SSH server takes effect.

```
Device# show ip ssh
SSH Enable - version 2.0
SSH Port:               22
SSH Cipher Mode:        ctr,gcm
SSH HMAC Algorithm:     sha2-256,sha2-512
Authentication timeout: 120 secs
Authentication retries: 3
SSH SCP Server:         disabled
SSH ip-block:           enabled
```

- Run the **show ssh** command to display the SSH connection information. Verify that a connection with username **hostname1** exists.

```
Device# show ssh
Connection Version Encryption       Hmac         Compress   State
Username
         1     2.0 aes256-ctr       hmac-sha2-256 none       Session started
hostname1
```

- Run the **show users** command to display information about connected clients. Verify that a client whose username is **hostname1** exists.

```
Device#show users
Line             User         Host(s)              Idle       Location
---------------- ------------ -------------------- ----------
------------------
   0 con 0       ---          idle                 00:00:16   ---
*  1 vty 0       hostname1    idle                 00:00:00   192.168.1.2
```

6. Configuration Files

```
!
username hostname1 password 7 $10$2fe$Rd/C4ijEl+UN$
```

```
!
enable service ssh-server
!
line console 0
line vty 0 4
 login local
!
```

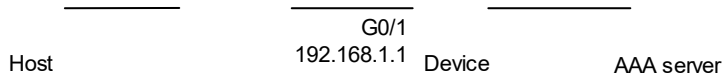
1.8.3 Configuring AAA Authentication for SSH Clients

1. Requirements

SSH clients can be authenticated using the AAA mode.

2. Topology

Figure 1-1 Configuring AAA Authentication for SSH Clients



3. Notes

- Configure the AAA server and add an AAA user on the AAA server.
- Enable the SSH server function.
- Enable the AAA service on the device and set the SSH authentication method to AAA server.
- Enable AAA authentication in the line.

4. Procedure

(1)Configure the AAA server and add an AAA user whose username is **hostname1** and password is **password1** on the AAA server. (For details about AAA server configuration, see the AAA server configuration guide.)

(2)Enable the SSH server function.

```
Device> enable
Device# configure terminal
Device(config)# enable service ssh-server
```

(3)Enable AAA services.

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

(4)Configure the address and shared key of the AAA server (a RADIUS server is used as an example here, the IP address is set to **192.168.1.3**, and the shared key is set to **radiuskey**).

```
Device(config)# radius-server host 192.168.1.3 key radiuskey
```

(5)Configure the login authentication method list **login-method** and select **radius** as the authentication method. The backup method **local** is configured. This ensures that a backup solution is available when the AAA server is faulty.

```
Device(config)# aaa authentication login login-method group radius local
```

(6)Enable AAA authentication in the line.

```
Device(config)# line vty 0 4
Device(config-line)# login authentication login-method
```

5. Verification

- Use the client software to log in to the device through SSH. The username is **hostname1**, and the password is **password1**. Verify that the login is successful. Use the PuTTY software as an example. The login page is as follows:

```
login as: hostname1
hostname1@192.168.1.1's password:



Last login: Apr  2 2021 10:00:00 from 192.168.1.2 through ssh.
Device>
```

- Run the **show ip ssh** command to verify that the SSH server takes effect.

```
Device# show ip ssh
SSH Enable - version 2.0
SSH Port:               22
SSH Cipher Mode:        ctr,gcm
SSH HMAC Algorithm:     sha2-256,sha2-512
Authentication timeout: 120 secs
Authentication retries: 3
SSH SCP Server:         disabled
SSH ip-block:           enabled
```

- Run the **show ssh** command to display the SSH connection information. Verify that a connection with username **hostname1** exists.

```
Device# show ssh
Connection Version Encryption       Hmac         Compress   State
Username
         1     2.0 aes256-ctr       hmac-sha2-256 none       Session started
hostname1
```

- Run the **show users** command to display information about connected clients. Verify that a client whose username is **hostname1** exists.

```
Device#show users
Line             User         Host(s)              Idle       Location
---------------- ------------ -------------------- ----------
------------------
   0 con 0       ---          idle                 00:00:16   ---
*  1 vty 0       hostname1    idle                 00:00:00   192.168.1.2
```

6. Configuration Files

```
!
aaa new-model
aaa authentication login login-method group radius local
!
```

```
radius-server host 192.168.1.3 key 7 $10$3b4$ln0s66i8XfEi$
!
line vty 0 4
 login authentication login-method
!
```

1.8.4 Configuring Public Key Authentication for SSH Clients

1. Requirements

When the device serves as the SSH server, users can use public key authentication to log in to the device without needing a password.

2. Topology

Figure 1-1 Configuring Public Key Authentication for SSH Clients

3. Notes

- Use the client software to generate a pair of keys (public key and private key).
- Copy the public key file to the flash memory of the device and associate the key with the user.

4. Procedure

(1)Configure a local user and enable local user authentication in a line. For details, see 1.8.2 Configuring Local User Authentication for SSH Clients.

(2)Use the client software to generate a pair of keys. The Ubuntu environment is used as an example. The generated private key is **id_rsa** and public key is **id_rsa.pub**.

```
root@ubuntu:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:6a/sWJjpx97QQEs9Aoiph5Ry452ybIyNyZUsT48Irrw root@ubuntu
The key's randomart image is:
+---[RSA 2048]----+
|  .o ..          |
|.o= .  . .       |
|o=.o..  + o      |
|+.+=o  o + .     |
|+O*oo   S        |
|++*o . = o       |
|o.    +.+ .      |
```

```
|..   . +o+       |
| E.   o+=.o      |
+----[SHA256]-----+
```

(3)Copy the public key file to the flash memory of the device (omitted).

(4)Associate the public key with **hostname1**.

```
Device> enable
Device# configure terminal
Device(config)# ip ssh peer hostname1 public-key rsa flash:test_key.pub
```

5. Verification

Verify that users can log in to the device through public key authentication. (The Ubuntu environment is used as an example.)

```
root@ubuntu:~/.ssh# ls
id_rsa  id_rsa.pub
root@ubuntu:~/.ssh# ssh -i id_rsa hostname1@192.168.1.1



Last login: Apr 2 2021 10:00:00 through console.
Device>
```

6. Configuration Files

```
!
username hostname1 password 7 $10$2fe$Rd/C4ijEl+UN$
!
ip ssh peer hostname1 public-key rsa flash:id_rsa.pub
!
enable service ssh-server
!
line console 0
line vty 0 4
 login local
!
```

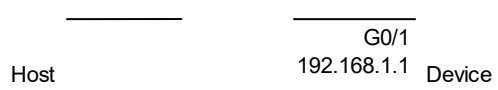
1.8.5 Configuring SCP Server

1. Requirements

When the device serves as the SCP server, users can run the **scp** command to upload files to and download files from the device.

2. Topology

Figure 1-1Configuring SCP Server



3. Notes

- Enable the SSH server function.
- Enable the SCP server function.

4. Procedure

(1)Configure a local user and enable local user authentication in a line. For details, see 1.8.2 Configuring Local User Authentication for SSH Clients.

(2)Enable the SSH server function.

```
Device> enable
Device# configure terminal
Device(config)# enable service ssh-server
```

(3)Enable the SCP server function.

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

5. Verification

Verify that the client can run the **scp** command to upload files to the device. The Ubuntu environment is used as an example here.

(1)In the Ubuntu shell environment, run the **scp** command to upload **file.txt** to the device.

```
root@ubuntu:~# scp file.txt hostname1@192.168.1.1:/
hostname1@192.168.1.1's password:
file.txt                                      100%   14     0.0KB/s   00:00
```

(2)Check whether **file.txt** exists in the flash directory of the device.

```
Device> enable
Device# dir flash:file.txt
Number  Properties     Size              Time                      Name
------  ----------  ----------  ------------------------  --------------------
  1       -rwx      14B       Fri Apr 2 10:00:00 2021  file.txt
```

Verify that the client can run the **scp** command to download files from the device. The Ubuntu environment is used as an example here.

(1) The **file.txt** file exists in the flash directory of the device.

```
Device> enable
Device# dir flash:file.txt
Number  Properties     Size              Time                      Name
------  ----------  ----------  ------------------------  --------------------
  1       -rwx      14B       Fri Apr 2 10:00:00 2021  file.txt
```

(3)In the Ubuntu shell environment, run the **scp** command to download **file.txt** from the device and rename it as **file1.txt**.

```
root@ubuntu:~# scp hostname1@192.168.1.1:/file.txt file1.txt
hostname1@192.168.1.1's password:
file.txt                                      100%   14     0.0KB/s   00:00
```

(4)In the Ubuntu shell environment, verify that the **file1.txt** file has been downloaded.

```
root@ubuntu:~# ls -l file1.txt
-rwxr--r-- 1 root root 14 Apr 2 10:00 file1.txt
```

6. Configuration Files

```
!
username hostname1 password 7 $10$2fe$Rd/C4ijEl+UN$
!
enable service ssh-server
!
ip scp server enable
!
line console 0
line vty 0 4
 login local
!
```

1.8.6 Configuring SSH Client

1. Requirements

When the device serves as an SSH client, the device can log in to the SSH server through SSH.

2. Topology

Figure 1-1Configuring SSH Client

3. Notes

- Configure a valid user on the SSH server.
- Run the **ssh** command to log in to the SSH server.

4. Procedure

(1)Configure a valid user on the SSH server. (For details, see the SSH server user manual.)

(2)Run the **ssh** command to log in to the SSH server.

```
Device> enable
Device# ssh -l admin 192.168.1.2
```

5. Verification

Verify that the device logs in to the SSH server successfully.

```
Device# ssh -l hostname1 192.168.1.2
%Trying 192.168.1.2, 22,...open
hostname1@192.168.1.2's password:
```

```
Last login: Apr 2 2021 10:00:00 from 192.168.1.1 through ssh.
```

1.8.7 Configuring SCP Client

1. Requirements

When the device serves as an SCP client, the device can run the **scp** command to upload files to and download files from the SCP server.

2. Topology

Figure 1-1 Configuring SCP Client

3. Notes

- Configure a valid user on the SCP server.
- Run the **scp** command to upload files to or download files from the SCP server.

4. Procedure

(1)Configure a valid user on the SCP server. For details, see the SCP server user manual.

(2)Run the **scp** command to copy **file.txt** from the SCP server to the flash memory of the device.

```
Device> enable
Device# scp hostname1@192.168.1.2:/file.txt flash:/file.txt
```

(3)Run the **scp** command to copy local file **file.txt** from the device to the SCP server.

```
Device> enable
Device# scp hostname1@192.168.1.2:/file.txt flash:/file.txt
```

5. Verification

- Verify that the file on the SCP server is successfully copied to a local directory.

```
Device> enable
Device# scp oob flash:/file.txt hostname1@192.168.1.2:/file.txt

%Trying 192.168.1.2, 22,...open
hostname1@192.168.1.2's password:
Press Ctrl+C to quit
!
Transmission success.
```

- Verify that the local file is successfully copied to the SCP server.

```
Device> enable
Device# scp hostname1@192.168.1.2:/file.txt flash:/file.txt
```

```
%Trying 192.168.1.2,...open
hostname1@192.168.1.2's password:
Press Ctrl+C to quit
!
Transmission success.
```