15.10.3 Public Key Authentication Facts

This lesson covers the following topics:

- Authenticate Using a Public Key
- Enable Public Key Authentication
- Creating self-signed certificates

Authenticate Using a Public Key

Public key authentication uses a public key instead of a username and password to authenticate an SSH connection.

The following method is used to authenticate using a public key:

1. The client specifies which public key the server uses for authentication. Then, the server checks to ensure the key has previously been authenticated to the server.
2. If the key is known to the server, it encrypts a random number with the public key and sends the encrypted number to the client.
3. The client decrypts the number with a private key and uses its own public key and the random number to create a hash (MD5 checksum). The client sends the hash back to the server.
4. The server uses the public key and the random number to create its own hash (MD5 checksum) and then checks whether both hash values match.
5. If the hashes match, the server grants access to the user. If the hashes do not match, the user is prompted to log in using a password.

A digital signature uses an asymmetric key pair to allow a sender's identity to be verified by a recipient. The sender creates the digital signature with a private key. The recipient decrypts the signature with the corresponding public key to verify the signature. The digital signature provides non-repudiation, meaning that the sender can not repudiate being the sender of the message. Digital signatures do not provide message integrity.

Public keys can also verify the integrity of messages sent. To ensure message integrity, the sender of a message creates a hash value for the message being transmitted. This hash value is called a message digest. The sender sends both the message digest and the message to the recipient. The recipient creates a hash for the message. If the recipient's hash matches the message digest, the integrity of the message is verified.

Enable Public Key Authentication

Use the following commands and files to enable public key authentication:

Command	Function	Example
/etc/ssh/sshd_config **etc/ssh/sshd.config**	Configures the server to accept public key authentication. Commonly used options for configuring a public key authentication on the server include: • **PubkeyAuthentication** enables and disables public key authentication on the server. • **AuthorizedKeysFile** *location* specifies the location of the file that contains the public keys.	**PubkeyAuthentication yes** enables public key authentication on the server. **AuthorizedKeysFile .ssh/authorized_keys** specifies the location of the file that contains the public keys.
ssh-keygen	Creates a key on the client to use when authenticating to a server. **ssh-keygen** options include: • **-t dsa** creates a DSA key pair (e.g., id_dsa and id_dsa.pub). • **-t rsa** creates an RSA key pair (e.g., id_rsa and id_rsa.pub).	**ssh-keygen -t dsa** creates a DSA key pair. **ssh-keygen -t rsa** creates an RSA key pair.
scp	Securely copies the client's public key file to the server.	**scp ~/.ssh/id_rsa.pub bjones@hs2.corpnet.com:/home/bjones/** copies id_rsa.pub to the home directory of bjones.
ssh	Logs in to the server.	**ssh -l bjones hs1** logs in to the hs1 computer as bjones.
cat	Appends the public key to the **~/authorized_keys** file. Be aware of the following: • Overwriting the file deletes all other keys.	**cat id_rsa.pub >> ~/.ssh/authorized_keys** appends the id_rsa.pub file to the end of the authorized_keys file.

	ⓘ Be sure to use **>>** instead of **>** when redirecting the output of the **cat** command. • If the same user logs in from multiple clients, the file must have all client keys in it. • Always remove the public key file after appending it to the **~/authorized_keys** file.	
ssh-agent bash **ssh-add**	Configures the client to automatically provide the private key passphrase when needed so that it does not have to be typed for every new connection to a server. 1. Use **ssh-agent bash** to enable passphrase agent. 2. Use **ssh-add** to specify the name of the private key to add to the agent. For protocol 2, this is one of the following: ◦ **~/.ssh/id_rsa** ◦ **~/.ssh/id_dsa** After the **ssh-add** command, enter the passphrase when prompted. The passphrase stays in memory while the user is logged in to the client.	**ssh-agent bash** enables passphrase automation. **ssh-add ~/.ssh/id_rsa** specifies the id_rsa file as the private key.
ssh-copy-id	Copies the public key of your default identity to the remote host using **ssh-copy-id user@hostname.com**. If you have only one ssh key, you do not have to entry your identity.	**ssh-copy-id** copies your default identity to the remote host. **ssh-copy-id user@hostname.com** copies the public key of the specified user.

Creating Self-signed Certificates

A self-signed certificate lets you encrypt communication between your server and a client but is not signed by any trusted certificate authorities. Therefore, users cannot use the certificate to validate the identity of your server automatically. Self-signed certificates are often used when you don't have domain name associated with your server where the encrypted web interface is not user-facing.

The exact steps to create a self-signed certificate will vary from distribution to distribution. The following is one example of how to create a self-signed certificate on Ubuntu. These steps may need to be modified for other distributions.

To create a self-signed certificate, open a terminal and as root or using sudo, run the following command:

openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/my-selfsigned.key -out /etc/ssl/certs/my-selfsigned.crt

Commands / Options	Description
openssl	The command used for creating and managing OpenSSL certificates.
req	Specifies that the X.509 certificate signing request (CSR) will be used. X.509 is a public key infrastructure standard that SSL and TLS adheres to for its key and certificate management.
-x509	Specifies that the certificate being created will be a self-signed instead of a signed certificate.
-nodes	Specifies that openssl should skip the option to secure our certificate with a passphrase. This allows a connection to the server (such as an Apache Web server) without user intervention.
-days *xxx*	Specifies the length of time that the certificate will be considered valid, where *xxx* is the number of days.
-newkey rsa:2048	Specifies that the a new certificate and new key will be created at the same time. The **rsa:2048** portion tells it to make an RSA key that is 2048 bits long.
-keyout	Specifies the directory and filename for the private key being created.
-out	Specifies the directory and filename for the self-signed key being created.

The following steps may also be required to use the self-signed certificate:

- Configure the server to use SSL.
- Adjust the firewall. If you are using the ufw firewall in the enabled mode, you might need to adjust the settings to allow for SSL traffic.
- Install the self-signed certificate in your server, such as the Apache Web server.
- Restart the server.
- Test the website with https.