SSH

2025-05-23
Tags: #Protocol
Related to: Footprinting
Author: Amr Zakaria

SSH (Secure Shell) is a cryptographic network protocol used for secure communication, remote command-line login, remote command execution, and other secure network services between two networked computers. It was developed in 1995 as a secure replacement for Telnet, rlogin, and other insecure protocols. SSH encrypts all traffic, including passwords, to prevent eavesdropping, connection hijacking, and other attacks.

Key Features of SSH

- Uses TCP port 22 by default
- Provides strong encryption (AES, ChaCha20), integrity checks (HMAC-SHA2), and host authentication
- Supports password-based and key-based authentication
- Allows secure file transfers (SFTP, SCP) and port forwarding
- Widely used for remote server administration, Git operations, and secure tunneling

---

How SSH Works

SSH operates in three main phases:

1. **Connection Establishment**
   - Client connects to the server on port 22
   - Server presents its host key for verification
2. **Key Exchange & Encryption Setup**
   - Client and server negotiate encryption algorithms
   - Establish a shared secret using Diffie-Hellman key exchange
3. **Authentication & Session**
   - User authenticates via password or public key
   - Encrypted communication begins

SSH Components

1. SSH Protocol Layers

Layer	Function
Transport Layer	Handles encryption, integrity, and server authentication
User Authentication Layer	Manages client authentication (password/key)
Connection Layer	Multiplexes multiple channels (shell, SFTP, tunnels)

2. SSH Key Files

File	Purpose
`~/.ssh/id_rsa`	Private key (keep secure!)
`~/.ssh/id_rsa.pub`	Public key (shared with servers)
`~/.ssh/known_hosts`	Stores verified host keys
`~/.ssh/authorized_keys`	Lists approved public keys for login

SSH Authentication Methods

1. Password Authentication

```
ssh username@hostname
# Prompts for password (encrypted during transmission)
```

Pros: Simple to set up
Cons: Vulnerable to brute-force attacks

2. Public Key Authentication (Recommended)

```
# Generate key pair
ssh-keygen -t ed25519

# Copy public key to server
ssh-copy-id username@hostname
```

Pros:

- More secure (resistant to brute force)

- Enables passwordless login
- Supports key passphrases for extra security

Common SSH Commands

Basic Connection

Command	Description
`ssh user@host`	Connect to host as user
`ssh -p 2222 user@host`	Connect to non-standard port
`ssh -i ~/.ssh/key.pem user@host`	Use specific private key

File Transfer

Command	Description
`scp file.txt user@host:/path`	Upload file via SCP
`scp user@host:/path/file.txt .`	Download file via SCP
`sftp user@host`	Interactive SFTP session

Port Forwarding

Command	Description	
`ssh -L 8080:localhost:80 user@host`	Local port forwarding	
`ssh -R 9000:localhost:3000 user@host`	Remote port forwarding	
`ssh -D 1080 user@host`	SOCKS proxy tunneling	

> SSH can create a **dynamic encrypted SOCKS proxy tunnel**, allowing you to securely route traffic from your local machine through a remote server.

SSH Server Configuration (sshd)

Configuration file: `/etc/ssh/sshd_config`

Security Best Practices

```
# Disable root login
PermitRootLogin no

# Restrict authentication methods
PasswordAuthentication no
PubkeyAuthentication yes

# Limit users/groups
AllowUsers admin deploy
AllowGroups ssh-users

# Advanced security
MaxAuthTries 3
LoginGraceTime 1m
UsePAM yes
X11Forwarding no
```

Restart SSH Service

```
sudo systemctl restart sshd
sudo systemctl enable sshd
```

SSH Hardening Guide

Step 1: Key-Based Authentication Only

1. Generate keys on client:

   ```
   ssh-keygen -t ed25519 -a 100
   ```

2. Copy public key to server:

   ```
   ssh-copy-id -i ~/.ssh/id_ed25519.pub user@server
   ```

3. Disable passwords in `/etc/ssh/sshd_config`:

```
   PasswordAuthentication no
```

Step 2: Firewall Configuration

```
sudo ufw allow 22/tcp
```

```
sudo ufw enable
```

Step 3: Fail2Ban Setup

Install and configure Fail2Ban to block brute-force attempts:

```
sudo apt install fail2ban
sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

Edit `/etc/fail2ban/jail.local`:

```
[sshd]
enabled = true
maxretry = 3
bantime = 1h
```

SSH Troubleshooting

Common Issues

Error	Solution
`Permission denied (publickey)`	Verify `authorized_keys` permissions (600)
`Host key verification failed`	Remove offending key from `known_hosts`
`Connection refused`	Check firewall/SSH daemon status
`Too many authentication failures`	Use `-o IdentitiesOnly=yes` with `-i`

Debug Mode

```
ssh -vvv user@host  # Verbose output
sudo journalctl -u sshd -f  # View server logs
```

SSH Security Scanning with Nmap

1. Basic SSH Detection

```
nmap -p 22 <target_IP>
```

2. SSH Version & Algorithms

```
nmap -p 22 --script=ssh2-enum-algos <target_IP>
```

3. Vulnerability Checks

```
# Check for weak algorithms
nmap -p 22 --script=ssh-auth-methods,sshv1 <target_IP>

# Check for legacy vulnerabilities
nmap -p 22 --script=ssh-run <target_IP>
```

4. Full SSH Audit

```
nmap -p 22 --script="ssh-*" -sV <target_IP>
```

Advanced SSH Features

1. SSH Config File (`~/.ssh/config`)

```
Host myserver
    HostName server.example.com
    User admin
    Port 2222
    IdentityFile ~/.ssh/myserver_key
    ForwardX11 yes
```

2. SSH Agent Forwarding

```
# Start agent
eval $(ssh-agent)

# Add key
ssh-add ~/.ssh/id_ed25519

# Use forwarding
ssh -A user@host
```

3. Multiplexing (Faster Connections)

```
# ~/.ssh/config
Host *
    ControlMaster auto
```

```
    ControlPath ~/.ssh/sockets/%r@%h-%p
    ControlPersist 600
```

SSH Best Practices

1. **Always use key authentication** - Disable password logins
2. **Use strong algorithms** - Prefer ed25519 over RSA
3. **Restrict access** - Use `AllowUsers` / `AllowGroups`
4. **Update regularly** - Patch against vulnerabilities
5. **Monitor logs** - Watch for brute force attempts
6. **Use bastion hosts** - For critical infrastructure access

> A bastion host (also called a jump server or jump host) is a specially configured server that acts as a secure gateway between untrusted networks (like the internet) and trusted internal networks. It's the **only** server exposed to the internet in a properly secured architecture.

7. **Disable unused features** - X11, port forwarding if not needed

---

SSH Client Tools

Tool	Purpose
OpenSSH	Standard Unix client (`ssh` , `scp` , `sftp`)
PuTTY	Popular Windows SSH client
MobaXterm	Enhanced Windows SSH with X11
Termius	Cross-platform SSH client
SecureCRT	Commercial SSH client