

Secure Shell - SSH



Secure Shell - SSH

- program that allows secure network services over an insecure network, such as the Internet
- Internet protocol that allows a user to connect to a remote host via an encrypted link
- program to securely log into another computer over a network
- to execute commands safely in a remote machine
- to securely copy/move files from one machine to another
- is intended as a replacement for insecure "Berkeley services":
 - telnet, rlogin, rsh, and rcp
- provides secure X connections over the network
- provide secure encrypted and authenticated communications between two hosts
- secure forwarding of arbitrary TCP connections/services

Secure Shell - SSH

- How do SSH, Telnet and Rlogin differ ?
 - SSH is a recently designed, high-security protocol
 - SSH uses strong cryptography to protect your connection against eavesdropping, hijacking and other attacks
 - Telnet and Rlogin are both older protocols offering minimal security
 - SSH and Rlogin both allow you to log in to the server without having to type a password
 - SSH allows you to connect to the server and automatically send a command
 - If you are connecting across the open Internet, then we recommend you use SSH
 - If you are behind a good firewall, it is more likely to be safe to use Telnet or Rlogin, but we still recommend you use SSH.

Secure Shell - SSH

- A typical SSH connection



Secure Shell - SSH

- Why should I use it ?
 - Traditional BSD 'r' - commands (rsh, rlogin, rcp) are vulnerable to different kinds of attacks
 - Somebody who has physical access to the wire, can gain unauthorized access to systems in a variety of ways
 - Protect against eavesdrop and logging of all the traffic to and from your system, including passwords
 - SSH offers strong host and user authentication methods
 - X Window System also has a number of severe vulnerabilities
- Powerful guardian against the numerous security hazards that nowadays threaten network communications

Secure Shell - SSH

- What kinds of attacks does SSH protect against ?
 - Eavesdropping a transmission - looking for passwords, credit card numbers, or business secrets
 - Hijacking - taking over a communication and redirect
 - Interception of communication between two systems - inspect or modify any data being transmitted thru itself
 - Impersonation of a particular host - an intercepting system pretends to be the intended recipient
 - IP spoofing, or faking network addresses or routing information
 - fool access control mechanisms
 - redirect connections to a fake server

All techniques cause information to be intercepted, possibly for hostile reasons

Secure Shell - SSH

- SSH – Secure Shell
 - **SSH never trusts the network !**
 - Can only force SSH to disconnect, but cannot decrypt or play back the traffic, or hijack the connection
- What kinds of attacks does SSH not protect against ?
 - Anything that compromises your host's security in some other way
 - Has gained root access to a machine, he can then subvert SSH
 - Has access to your home directory, then security is nonexistent



Secure Shell - SSH

- What software packages are available for implementing SSH?

Client / Servers:

- OpenSSH – www.openssh.org - open source
- fressh – www.fressh.org - open source
- SSH Secure Shell – www.ssh.com - commercial
- F-secure – www.f-secure.com - commercial
- PuTTY – www.chiark.greenend.org.uk/~sgtatham/putty/ - free

OpenSSH



- Open Source Project
- Free Licensing
- Strong Encryption (3DES, Blowfish)
- X11 Forwarding (encrypt X Window System traffic)
- Port Forwarding (encrypted channels for legacy protocols)
- Strong Authentication (Public Key, One-Time Password)
- Agent Forwarding (Single-Sign-On)
- Interoperability (Compliance with SSH 1.3, 1.5, and 2.0 protocol Standards)
- SFTP client and server support in both SSH1 and SSH2 protocols.
- Kerberos and AFS Ticket Passing
- Data Compression

OpenSSH Server configuration

Install the **openssh-server** and **openssh** rpm packages

- `/usr/sbin/sshd` – SSH server daemon
- `/etc/ssh/sshd_config` – server configuration file
- The default config file is sufficient
- `/sbin/service sshd start` – to start the SSH service
- `/sbin/service sshd stop` – to stop the SSH service
- `/sbin/service sshd restart` – to restart the SSH service

- OpenSSH packages also require the `openssl` package, which installs several important cryptographic libraries that help OpenSSH provide encrypted communications

OpenSSH Server configuration file

Edit the default config file: `# vi /etc/ssh/sshd_config`

- `Port 22`
 - Specifies the port number that **sshd** listens on
- `Protocol 2,1`
 - Specifies the protocol versions **sshd** supports
- `ListenAddress 0.0.0.0`
 - Specifies the local addresses **sshd** should listen on
- `HostKey /etc/ssh/ssh_host_rsa_key`
 - Specifies a file containing a private host key used by SSH
- `LoginGraceTime 600`
 - Specifies the time limit for a user to log in
- `PermitRootLogin yes`
 - Specifies whether root can login using ssh
- `StrictModes yes`
 - Specifies whether **sshd** should check file modes and ownership of the user's files and home directory before accepting login

OpenSSH Server configuration file:

- `PubkeyAuthentication yes`
 - Specifies whether public key authentication is allowed
- `PasswordAuthentication yes`
 - Specifies whether password authentication is allowed
- `PermitEmptyPasswords no`
 - When password authentication is allowed, it specifies whether the server allows login to accounts with empty password strings
- `MaxStartups 10`
 - Specifies the maximum number of concurrent unauthenticated connections to the **sshd** daemon
- `KeepAlive yes`
 - Specifies whether the system should send TCP keepalive messages to the other side
- `VerifyReverseMapping no`
 - Specifies whether **sshd** should try to verify the remote host name
- `Subsystem sftp /usr/libexec/openssh/sftp-server`

OpenSSH Client configuration

- Install **openssh-clients** and **openssh** packages on the client
- Install the desired Windows SSH client software
- `/etc/ssh_config` - system wide ssh client configuration file
- `~/.ssh` - user custom configuration file
- Configuration data is parsed as follows:
 - 1. command line options
 - 2. user-specific file
 - 3. system-wide file

OpenSSH commands

- `ssh`
 - The basic rlogin/rsh-like client program
- `sshd`
 - The daemon that permits you to login
- `ssh-agent`
 - An authentication agent that can store private keys
- `ssh-add`
 - Tool which adds keys to in the above agent
- `sftp`
 - FTP-like program that works over SSH1 and SSH2 protocol
- `scp`
 - File copy program that acts like rcp
- `ssh-keygen`
 - Key generation tool
- `sftp-server`
 - SFTP server subsystem (started automatically by sshd)

OpenSSH Port Forwarding

- Secure otherwise insecure TCP/IP protocols via port forwarding
- the SSH server becomes an encrypted conduit to the SSH client
- mapping a local port on the client to a remote port on the server
- the mapped port numbers do not need to match for it to work
- To create a TCP/IP port forwarding channel which listens for connections on the localhost, use the following command:

```
ssh -L local-port:remote-host:remote-port  
user@remote-host
```

Note: Setting up port forwarding to listen on ports < 1024 requires root access

OpenSSH Port Forwarding – type I



To check your email on a server called mail.domain.com using POP through an encrypted SSH connection to the POP server, you can use the following command:

```
ssh -L 1100:mail.domain.com:110 mail.domain.com
```

Once the port forwarding channel is in place between the two machines, you can **direct** your **POP mail client** to use port **1100 on localhost** to check for new mail. Any requests sent to port 1100 on your system will be directed **securely** to the mail.domain.com server's port 110.

OpenSSH Port Forwarding – type II



If mail.domain.com is not running an SSH server daemon, you can log in via SSH to a machine on the same network and still use SSH to secure a part of the POP connection.

```
ssh -L 1100:mail.domain.com:110 ssh-server.domain.com
```

You are forwarding your POP request from port 1100 on your machine through SSH connection on port 22 to ssh server.domain.com. Then, ssh-server.domain.com connects to port 110 on mail.domain.com to allow you to check for new mail. Only the connection between your system and ssh-server.domain.com is secure.

OpenSSH Lab

- OpenSSH server configuration
- Edit/view the sshd_config file
- Managing the sshd service
- OpenSSH Client configuration
- Using the ssh command with password authentication
- Using the ssh command with public key authentication
- Using the scp command
- Using the sftp command
- Port forwarding