DEFINITION

Secure Shell (SSH)

Posted by: Margaret Rouse WhatIs.com

Contributor(s): Peter Loshin, Michael Cobb

SSH, also known as Secure Shell or Secure Socket Shell, is a network protocol that gives users, particularly system administrators, a secure way to access a computer over an unsecured network. SSH also refers to the suite of utilities that implement the SSH protocol. Secure Shell provides strong authentication and encrypted data communications between two computers connecting over an open network such as the internet. SSH is widely used by network administrators for managing systems and applications remotely, allowing them to log into another computer over a network, execute commands and move files from one computer to another.

SSH refers both to the cryptographic network protocol and to the suite of utilities that implement that protocol. SSH uses the client-server model, connecting a secure shell client application, the end at which the session is displayed, with an SSH server, the end at which the session runs. SSH implementations often include support for application protocols used for terminal emulation or file transfers. SSH can also be used to create secure tunnels for other application protocols, for example, to securely run X Window System graphical sessions remotely. An SSH server, by default, listens on the standard Transmission Control Protocol (TCP) port 22.

While it is possible to use SSH with an ordinary user ID and password as credentials, SSH relies more often on public key pairs to authenticate hosts to each other. Individual users must still use their user ID and password (or other authentication methods) to connect to the remote host itself, but the local machine and the remote machine authenticate separately to each other. This is accomplished by generating a unique public key pair for each host in the communication; a single session requires two public key pairs: one public key pair to authenticate the remote machine to the local machine, and a second public key pair to authenticate the local machine to the remote machine.

SSH connections have been used to secure many different types of communications between a local machine and a remote host, including secure remote access to resources, remote execution of commands, delivery of software patches and updates and other administrative or management tasks.

Secure Shell capabilities

Functions that SSH enables include:

- Secure remote access to SSH-enabled network systems or devices, for users as well as automated processes;
- secure and interactive file transfer sessions;
- automated and secured file transfers;
- secure issuance of commands on remote devices or systems; and
- secure management of network infrastructure components.

SSH can be used interactively to enable terminal sessions, and should be used instead of the less secure Telnet program. SSH is also commonly used in scripts and other software to enable programs and systems to remotely and securely access data and other resources.

History of SSH

The first version of SSH appeared in 1995 and was designed by Tatu Ylönen, who was, at the time, a researcher at Helsinki University of Technology and later went on to start SSH Communications Security, a cybersecurity vendor based in Waltham, Mass. Over time various flaws have been found in SSH-1, and that version is now considered to be deprecated and not safe to use.

SSH-2, the current version of Secure Shell protocols, was adopted as a Standards Track specification by the Internet Engineering Task Force (IETF) in 2006. SSH-2 is not compatible with SSH-1 and uses a Diffie-Hellman key exchange and a stronger integrity check that uses message authentication codes to improve security. SSH clients and servers can use a number of encryption methods, the mostly widely used being Advanced Encryption Standard (AES) and Blowfish.

As yet, there are no known exploitable vulnerabilities in SSH-2, though information leaked by Edward Snowden in 2013 suggests the National Security Agency may be able to decrypt some SSH traffic.

Instead of passwords, SSH authenticates with public key cryptography.

How SSH works

Secure Shell was created to replace insecure terminal emulation or login programs such as Telnet, rlogin (remote login) and rsh (remote shell); SSH enables the same functions (logging into and running terminal sessions on remote systems). SSH also replaces file transfer programs such as File Transfer Protocol (FTP) and rcp (remote copy).

The most basic use of SSH is for connecting to a remote host for a terminal session. The form of that command is:

```
ssh UserName@SSHserver.example.com
```

This command will cause the client to attempt to connect to the server named `server.example.com`; using the user ID `UserName`. If this is the first time negotiating a connection between the local host and the server, the user will be prompted with the remote host's public key fingerprint and prompted to connect, despite there having been no prior connection:

```
The authenticity of host 'sample.ssh.com' cannot be established.
DSA key fingerprint is 01:23:45:67:89:ab:cd:ef:ff:fe:dc:ba:98:76:54:32:10.
Are you sure you want to continue connecting (yes/no)?
```

Answering "yes" to the prompt will cause the session to continue and the host key is stored in the local system's `known_hosts` file. This is a hidden file, stored by default in a hidden directory, called `/.ssh/known_hosts`, in the user's home directory. Once the host key has been stored in the `known_hosts` file, the client system can connect directly to that server again without need for any approvals: the host key authenticates the connection.

Secure Shell security issues

Enterprises using SSH should consider finding ways to manage host keys stored on client systems; these keys can accumulate over time, especially for IT staff who need to be able to access remote hosts for management purposes. Because the data stored in an SSH `known_hosts` file can be used to gain authenticated access to remote systems, organizations should be aware of the existence of these files and should have a standard process for retaining control over the files, even after a system is taken out of commission, as the hard drives may have this data stored in plain text.

Developers should also be careful when incorporating SSH commands or functions in a script or other type of program. While it is possible to issue an SSH command that includes a user ID and password to authenticate the user of the local machine to an account on the remote host, doing so may expose the credentials to an attacker with access to the source code.

Shellshock, a security hole in the Bash command processor, can be executed over SSH but is a vulnerability in Bash, not in SSH. In reality, the biggest threat to SSH is poor key management. Without the proper centralized creation, rotation and removal of SSH keys, organizations can lose control over who has access to which resources and when, particularly when SSH is used in automated application-to-application processes.

SSH vs. Telnet

Telnet was one of the first internet application protocols -- the other is the FTP -- and Telnet is used for initiating and maintaining a terminal emulation session on a remote host.

SSH and Telnet are functionally similar, with the primary difference between them being that the SSH protocol uses public key cryptography to authenticate end points when setting up a terminal session as well as for encrypting session commands and output.

While Telnet is primarily used for terminal emulation, SSH can be used to do terminal emulation (similar to the rlogin command) as well as for issuing commands remotely as with rsh, transferring files using SSH File Transfer Protocol (SFTP) and for tunneling other applications.

SSH vs. SSL/TLS

The Transport Layer Security (TLS) protocol, which updates the Secure Sockets Layer (SSL) protocol, was designed to provide security for network transmissions at the transport layer. The SSH protocol also operates at or just above the transport layer, but there are important differences between the two protocols.

While both rely on public/private key pairs to authenticate hosts, under TLS only the server is authenticated with a key pair. SSH uses a separate key pair to authenticate each connection: one key pair for a connection from a local machine to a remote machine, and a second key pair to authenticate the connection from the remote machine to the local machine.

Another difference between SSH and TLS is that TLS allows connections to be encrypted without authentication, or authenticated without encryption; SSH encrypts and authenticates all connections.

SSH provides IT and information security professionals with a secure mechanism for managing SSH clients remotely. Rather than requiring password authentication to initialize a connection between an SSH client and server, SSH authenticates the devices themselves. This enables IT staff to connect with remote systems and modify SSH configurations, including adding or removing host key pairs in the `known_hosts` file.

SSH implementations

As an open protocol, SSH has been implemented for most computing platforms and the open source OpenSSH implementation is the one most commonly found on Linux, Unix and other operating systems based on Berkeley Software Distribution (BSD), including Apple's MacOS.

OpenSSH was ported to run in Windows Power Shell starting in 2015, and in 2018 optional OpenSSH support was added to Windows 10. While SSH is directly accessible by default in most Unix-like operating systems, Microsoft's ported version of OpenSSH must be explicitly enabled in the Windows settings app.

PuTTY is another open source implementation of SSH, and while it currently is available for Windows, MacOS and Unix/BSD, PuTTY was originally written to run on Windows and it has long been one of the top options for using SSH on a Windows system.

Most implementations of the SSH suite comprise three utilities -- slogin, ssh and scp -- that are secure versions of the earlier insecure Unix utilities, rlogin, rsh and rcp. SSH uses public-key cryptography to authenticate the remote computer and allow the remote computer to authenticate the user, if necessary.

There are currently dozens of SSH implementations available for various platforms and under a variety of open source and proprietary licenses.

SSH commands

While there are graphical implementations of SSH, the program is usually invoked at the command line -- or executed as part of a script. Running the ssh command on its own, with no arguments such as a destination host or user ID, returns a list of SSH command parameters and options.

The most basic form of SSH command is to invoke the program and the destination host name or IP address:

```
ssh server.example.org
```

This will connect to the destination, `server.example.org`; the destination host will respond by prompting for a password for the user ID of the account under which the client is running. In other words, if the user ID in use is "jsmith", then the remote host will ask for a password associated with the account "jsmith" on the remote host. In many cases, the user ID for the remote host will be different, in which case the command should be issued with the remote host user ID, like this:

```
ssh remote_host_userID@server.example.org
```

SSH can also be used from the command line to issue a single command on the remote host, and then exit. For example:

```
ssh example.org ls
```

This command executes the Unix `ls` command, which lists all contents of the current directory on the remote host. While this example is trivial, it demonstrates that SSH can be used to execute more interesting commands on a remote host. For example, a command can be crafted that initializes a server instance that will give a particular remote machine access to a single file (or other resource), and then terminate the server after the file is accessed by the specified remote host.

In addition to the `ssh` executable, SSH has other executable commands used at the command line for additional functions, including:

- `sshd` initiates the SSH server, which waits for incoming SSH connection requests and enables authorized systems to connect to the local host.
- `ssh-keygen` is a program for creating a new authentication key pairs for SSH, which can be used to automate logins, single sign-on and to authenticate hosts.
- `ssh-copy-id` is a program used to copy, install and configure an SSH key on a server to automate passwordless logins and single sign-ons.
- `ssh-agent` is a "helper" program that tracks identity keys and their passphrases -- from which SSH derives an encryption key -- and enables the user to use the identity keys to log into different servers without the need to re-enter passwords or passphrases.
- `ssh-add` is used to add a key to the SSH authentication agent and is used with `ssh-agent` to implement single sign-on using SSH.

- `scp` is a program used for copying files from one computer to another and is an SSH-secured version of `rcp`, which is the Unix "remote copy" command.
- `sftp` is a program used for copying files from one computer to another and is an SSH-secured version of ftp, the original File Transfer Protocol. SFTP has become the preferred mechanism for file sharing over the internet, replacing both FTP and FTP/S (FTP with Security), which is a protocol for using FTP over an SSL/TLS tunnel.

SSH tunneling

SSH tunneling, also known as SSH port forwarding, is a technique that enables a user to open a secure tunnel between a local host and a remote host.

SSH port forwarding is a technique for redirecting network traffic to a particular port/IP address, so that a remote host is made directly accessibly by applications on the local host. The destination may be on the remote SSH server, or that server may be configured to forward to yet another remote host.

SSH tunnels are powerful tools for IT administrators as well as malicious actors because they are capable of transiting an enterprise firewall undetected. As a result, there are tools available to prevent unauthorized use of SSH tunnels through a corporate firewall.

This was last updated in October 2018

Continue Reading About Secure Shell (SSH)

- Learn more about SSH security in the cloud
- Take steps to improve SSH security in the enterprise
- Protect yourself against SSH brute force attacks
- Thwart SSH attacks on a network's nonstandard ports
- OpenBSD man pages and specifications for ssh and SSH2

Related Terms

one-time password (OTP)

A one-time password (OTP) is an automatically generated numeric or alphanumeric string of characters that authenticates the user ... See complete definition

remote access

Remote access is the ability to access a computer or a network remotely through a network connection. See complete definition

two-factor authentication (2FA)

Two-factor authentication (2FA), sometimes referred to as two-step verification or dual factor authentication, is a security ... See complete definition