SSH: Secure Shell Protocol

YASIR SAID ALHANINI
Department of Information Science and Technology
Universiti Kebangsaan Malaysia
Bangi, Malaysia
$yas_7000@hotmail.com$

Abstract

Information when sent in plaintext leaves them wide open to interception as most networks tend to be insecure to a large extent. In 1995, the first version of the protocol (now called SSH-1) was designed as a result of password-sniffing attack at a university network in Finland. Secure Shell (SSH) is a network protocol that allows data to be exchanged using a secure channel between two networked devices. SSH was designed as a replacement for Telnet and other insecure remote shells. Its uses encryption and public key cryptography for confidentiality, integrity and authentication of data and can be used for many applications. This paper is aimed at reviewing the different applications, uses, advantages and failings of SSH as an internet tool.

1. Introduction

Secure Shell is a program to log into another computer over a network, to move files from one machine to another and to execute commands in a remote machine. SSH provides a very strong authentication and secure communications over insecure channels. It also provides secure X connections and secure forwarding of arbitrary TCP connections. Secure Shell can also be used as a tool for things like rsync(an open source utility that provides fast incremental file transfer) and secure network backups. It is a network procedure that provides a replacement for insecure remote login and command execution facilities like; telnet, rlogin, rsh, and rcp and provide secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel. SSH prevents password sniffing and password theft by encrypting traffic in both directions. It also performs the following functions

a. Public Key Authentication: optionally replace password authentication.

b. Authentication of the server thereby making man-in-the-middle attack impossible.

c. X11 forwarding.

d. File transfer as it include two file transfer protocols

e. Port forwarding: TCP sessions can be forwarded over an SSH connection.

f. Compression: Traffic may be optionally compressed at the stream level.

The traditional BSD 'r' - commands (rsh, rlogin, rcp) are vulnerable to different kinds of attacks. A hostile person who has root access to machines on the network, or physical access to the wire, can gain unauthorized access to systems in a variety of ways. It is also possible for such a person to log all the traffic to and from your system, including passwords (which ssh does not send in the clear). SSH computes new session keys whenever a new SSH session is established. This computationally expensive operation causes significant latency and processor load on low end devices and thus makes short-lived connections very expensive The X Window System also has a number of severe vulnerabilities. With SSH, you can create secure remote X sessions which are transparent to the user. As a side effect, using remote X clients with SSH is more convenient for users.

2. VERSIONS OF SSH

There are two versions of Secure Shell available namely SSH1 and SSH2. They are two entirely different protocols that are not compatible with each other. They encrypt at different parts of the packets.

SSH1: Uses server and host keys to authenticate systems. SSH1 has more supported platforms, supports .rhosts authentication (against the draft for SSH2) and has more diverse authentication support (AFS, Kerberos, etc.).

SSH2: This only uses host keys. It is a complete rewrite of the protocol and does not use the same networking implementation that SSH1 does. It has secure terminal sessions utilizing secure encryption, secure replacement for FTP and Telnet, as well as the UNIX r-series of commands: rlogin, rcp, rexec, multiple high security algorithms and strong authentication methods that prevents security threats such as identity spoofing. SSH2 also possesses multiple ciphers for encryption, has transparent and automatic tunnelling of X11 connections and arbitrary TCP/IP-based applications, such as e-mail. It does automatic and secure authentication of both ends of the connection and both the server and the client are authenticated to prevent identity spoofing, Trojan horses, and so on. Lastly, it has enhanced multiple channels that allow multiple terminal windows and file transfers through one secure and authenticated connection. In other word, SSH2 is a rewrite of the SSH1 protocol with improvements to security, performance, and portability.

3. HOW SECURE SHELL WORKS

SSH works by the exchange and verification of information using private and public keys, to identify users and hosts. Also by using the public/private key cryptography, it then provides encryption of subsequent communication. SSH has two parts: a client and a server.

Client means a workstation or PC that that the user is already logged into (personal workstation or a group workstation that provides XDM session management for several X terminals). It is where the user types in "rlogin server" or "rcp file server:newfile"

Server means a secondary remote workstation that the user wishes to log into to do some work; a login session server. This is where you get a new login session and shell prompt or are copying files respectively.

User generates an "identity" on the client system by running the ssh-keygen program. This program creates a subdirectory HOME/ssh and inserts in it two files named identity and identity.pub which contain the users private and public keys for his/her account on the client system. This latter file can then be appended to a file HOME/ssh/authorized-keys that should reside on any/all servers where you will make ssh connections.

4. HOW SECURE SHELL AUTHENTICATES

Secure Shell authenticates using one or more of the following: Password (the /etc/passwd or /etc/shadow in UNIX): The password authentication method is the easiest to implement, as it is set up by default. Password authentication uses the /etc/passwd or /etc/shadow file on a UNIX system, depending on how the passwords are set up. To enable password authentication, users must make sure that the AllowedAuthentications field of the /etc/ssh2/sshd2-config and /etc/ssh2/ssh2-config configuration files contain the parameter password: AllowedAuthentications password

User public key (RSA or DSA, depending on the release): Instead of authenticating the user with a password, the server will verify a challenge signed by the users private key against its copy of the users public key. Setting up a public key authentication requires the generation of a public/private key pair and installing the public portion on the server.

Kerberos (for SSH1): SSH Secure Shell only supports Kerberos5. When Kerberos support is enabled, it is possible to authenticate using Kerberos credentials, forwardable TGT (ticket granting ticket) and passing TGT to remote host for single sign-on.

Host-based (.rhosts or /etc/hosts.equiv in SSH1 or public key in SSH2)

Server Authentication: The remote host can authenticate itself using either traditional public-key authentication or certificate authentication. At the beginning of the connection the server sends its public host key to the client for validation. If certificate authentication is used the public key is included in the certificate the server sends to the client.

Certificate Authentication: In this instance, the server sends its certificate (which includes its public key) to the client. As the server certificate is signed with the private key of a certification authority (CA), the client can verify the validity of the server certificate by using the CA certificate. The client then checks that the certificate contains the fully qualified domain name of the server. The client verifies that the server has a valid private key by using a challenge. When certificates are used, a man-in-the-middle attack is not a threat during key exchange, because the system checks that the server certificate has been issued by a trusted certifying authority (CA).

Since there is quite a big demand for SSH, there are some patches available for various forms of authentication.

5. COMPONENTS OF SECURE SHELL

The SSH protocol consists of three major components:

a. The Transport Layer Protocol: This provides server authentication, confidentiality, and integrity with perfect forward secrecy.SSH transport protocol provides a confidential channel over an insecure network. It performs encryption, server host authentication, integrity protection and key exchange. It also derives a unique session id that may be used by higher-level protocols.

b. The User Authentication Protocol authenticates the client to the server. Provides a set of mechanisms which are used to authenticate the client user to the server. The individual mechanisms in the set make use of the session IDs provided by the transport protocol and depends on the integrity and security guarantees of the transport protocol. Authentication is client-driven: when one is prompted for a password, it may be the SSH client prompting, not the server. The server merely responds to client's authentication requests.

c. The Connection Protocol multiplexes the encrypted tunnel into several logical channels. It also make specifications on a mechanism to multiplex multiple channels of data over the confidential and authenticated transport. It also specifies streams for accessing an interactive shell, for 'proxy-forwarding' various external protocols over the secure transport (including arbitrary TCP/IP protocols), and for accessing secure 'subsystems' on the server host. A single

SSH connection can host multiple channels simultaneously, each transferring data in both directions. Channel requests are used to relay out-of-band channel specific data, such as the changed size of a terminal window or the exit code of a server-side process. The SSH client requests a server-side port to be forwarded using a global request.

6. WHAT SECURE SHELL PROTECTS AGAINST

Secure Shell actively protect against the following:

Manipulation of data by people in control of intermediate hosts.

Interception of cleartext passwords and other data by intermediate hosts.

IP spoofing, where a remote host sends out packets which pretend to come from another, trusted host. SSH even protects a user against a spoofer on the local network, who can pretend he/she is the users router to the outside.

IP source routing, where a host can pretend that an IP packet comes from another, trusted host.

DNS spoofing, where an attacker forges name of server records.

Attacks based on listening to X authentication data and spoofed connection to the X11 server.

In one word, secure shell does not trust the net; if a hostile person takes over the network, he/she can only force secure shell to disconnect, but cannot decrypt, play back the traffic or hijack the connection. The above only holds if the user uses encryption. Secure Shell does have an option to use encryption of type ”none”. This is only for debugging purposes, and should NEVER be used!!!

On the other hand, Secure Shell will not assist the user with anything that compromises his host’s security in any way. Once an attacker gains root access to a machine, he can then subvert SSH too. If a hostile person has access to the users home directory then, there is no longer any security. This often happen if the home directory is exported via NFS.

7. SETTING UP SSH

Public Key distribution: To use SSH, the user has to authenticate his/herself to the remote host and the remote host in return authenticates itself to the user. The authentication is done by encrypting a message with the users private key. This is something only the user can do. The remote host decrypts the message with the public key and then knows that the user sent the message. The remote host daemon then encrypts a message with its private key, and the relevant client decrypts the message with the hosts public key. The client then knows that the host is really the host. User set up of SSH basically consists of sending the users public key to the host and getting the hosts public key.

In an ideal situation, there would be a public key server; a system where anybody could go and get the users public key. In the absence of such a system, the public key has to be distributed by hand.

8. USING SECURE SHELL

Once the user has validated himself/herself to the remote machine, he/she simply gives the command ssh remote$_machine$

9. SSH FORWARDING

As explained above, SSH protocol has the ability to multiplex various and diverse connections over a SSH channel. These is known as forwarding and it allows the user to transport TCP/IP, X11 and ssh-agent sessions over a SSH session. Some examples of forwarding are explained below;

a. X11 forwarding: The SSH protocol can forward X11 connections thereby allowing the user to securely display remote applications locally without the fear of any threat. It requires that the server end have an xauth binary accessible to set up the required authentication for the users server. For security reasons, X11 forwarding is not always on by default.

b. Port Forwarding: This allows SSH to forward arbitrary TCP sessions. The connections are fully encrypted since they are carried over a SSH tunnel thereby making port forwarding an additional security to traditionally insecure protocols. SSH supports port-forwarding from client to server and vice-versa. Using port-forwarding over a connection to the firewall, a user can easily gain access to all the TCP services of the protected machine as if he is connecting form the firewall itself.

c. Dynamic Port-forwarding: Dynamic port forwarding is a transparent mechanism available for applications which support the SOCKS4 or SOCKS5 client protocol. Instead of configuring port forwarding from specific ports on the local host to specific ports on the remote server, the user specifies a SOCKS server which can be used by his applications. This mode is used for burrowing through firewalls.

d. Authentication Agent Forwarding: This allows the user to forward a connection to a local ssh-agent so that the user can continue to use it on the remote machine. This may lead to security problems if switched on by default especially if forwarded to an untrusted host. Agent forwarding is very useful between trusted hosts.

10. SSH IMPLEMENTATION

OpenSSH is the most popular of the SSH implementations. It is a FREE version of the SSH connectivity tools that technical users of the Internet rely on. It supports both

SSH protocols and encrypts all traffic (including passwords) to effectively eliminate eavesdropping, connection hijacking, and other attacks. Additionally, OpenSSH provides secure tunneling capabilities and several authentication methods, and supports all SSH protocol versions. The OpenSSH suite replaces rlogin and telnet with the ssh program, rcp with scp, and ftp with sftp. Also included is sshd (the server side of the package), and the other utilities like ssh-add, ssh-agent, ssh-keysign, ssh-keyscan, ssh-keygen and sftp-server. Open SSH is now available in most free operating system distributions as well as many commercial ones. OpenSSH is developed by the OpenBSD Project (http://www.openbsd.org/). The software is developed in countries that permit cryptography export and is freely useable and re-useable by everyone under a BSD license.

11. Conclusion

SSH1 is now generally considered obsolete and should be avoided by explicitly disabling fall-back to it as it has inherent design flaws which makes it vulnerable it. While most modern clients and servers support SSH2, some organizations still use software with no support for SSH2, and thus SSH1 cannot always be avoided. In all versions of SSH, it is important to verify unknown public keys before accepting them as valid. Accepting an attacker's public key as a valid public key has the effect of disclosing the transmitted password and allowing man-in-the-middle attacks.

References

[1] http://www.ssh.fi. The company started by Tatu Ylonen (ssh's author) to develop and market ssh.

[2] http://www.openssh.com. The freely available version.

[3] ftp://ftp.zedz.net/pub/crypto/linux/. The offshore site that has carried encryption software for the Linux community for a long time.

[4] ftp://ftp.redhat.de/ /pub/rh-addons/security/. The offshore ftp site that carries Redhat's rpms.

[5] http://violet.ibs.com.au/openssh/. Linux port of OpenSSH.

[6] SSH: The Secure Shell Definitive Guide 2E

[7] SSH FAQ

[8] OPENSSH Project Official Site

[9] SSH Communications Security

[10] Michael Stahnke Pro OpenSSH, Apress 2005

[11] Daniel J. Barrett, Richard E. Silverman and Robert G. Byrnes SSH: The Secure Shell (The Definitive Guide), O'Reilly 2005 (2nd edition).

[12] Himanshu Dwivedi. Implementing SSH strategies for optimizing the Secure Shell, 2004

[13] Himanshu Dwivedi; Implementing SSH, Wiley 2003.

[14] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell(SSH)Protocol Architecture", January 2006.

[15] Orman, H., "The OAKLEY Key Determination Protocol", November 1998.

[16] Harkins, D. and D. Carrel, "The Internet Key Exchange(IKE)", November 1998.

[17] Nicholas Rosasco and David Larochelle. "How and Why More Secure Technologies Succeed in Legacy Markets: Lessons from the Success of SSH". Quoting Barrett and Silverman, SSH, the Secure Shell: The Definitive Guide, O'Reilly Associates (2001). Dept. of Computer Science, Univ. of Virginia. http://www.cs.virginia.edu/ drl7x/sshVsTelnetWeb3.pdf.

[18] http://www.javvin.com/protocol/sshdraft15.pdf:SSH Protocol Architecture

[19] http://www.javvin.com/protocol/sshtransport17.pdf:SSH Transport Layer Protocol

[20] http://www.javvin.com/protocol/sshauth18.pdf:SSH User Authentication Protocol

[21] http://www.javvin.com/protocol/sshconnect18.pdf:SSH Connection Protocol