IJCSMS International Journal of Computer Science & Management Studies, Vol. 11, Issue 02, Aug 2011
ISSN (Online): 2231 –5268
www.ijcsms.com

"A STUDY ON SECURE SHELL (SSH) PROTOCOL"

NIDHI KANDHIL[1] , Dr. ANIL KUMAR[2]
[1]***Research Scholar***, CMJ University, Shillong (Meghalaya)
[2]**Asstt. Professor & Head, Dept. of Computer Science & Applications,**
Pt. N.R.S. Govt. College, Rohtak (India)

ABSTRACT

Secure Shell (SSH) provides an open protocol for securing network communications that is less complex and expensive than hardware-based VPN solutions. Secure Shell client/server solutions provide command shell, file transfer, and data tunneling services for TCP/IP applications. SSH connections provide highly secure authentication, encryption, and data integrity to combat password theft and other security threats. VanDyke Software® clients and servers are mature native Windows implementations that offer a range of SSH capabilities and are interoperable with SSH software on other platforms.

Key words: SSH, Vshell Server, SecureCRT, SecureFX, OPEN SSH, OSSH, FreeSSh, GPL SSH

I. INTRODUCTION

SSH, the Secure Shell, is a popular software-based approach to network security. It is a protocol that allows user to log into another computer over a network, to execute commands in a remote machine, and to move files from one machine to another. It provides authentication and encrypted communications over unsecured channels.
The Secure Shell protocol provides four basic security benefits:

- **User Authentication**

- **Host Authentication**

- **Data Encryption**

- **Data Integrity**

Secure Shell authentication, encryption and integrity ensure identities and keep data secure.

User Authentication

Authentication, also referred to as user identity, is the means by which a system verifies that access is only given to intended users and denied to anyone else. Many authentication methods are currently used, ranging from familiar typed passwords to more robust security mechanisms.

Host Authentication

A host key is used by a server to prove its identity to a client and by a client to verify a "known" host. Host keys are described as persistent (they are changed infrequently) and are asymmetric—much like the public/private key pairs discussed above in the Public key section. If a machine is running only one **SSH** server, a single host key serves to identify both the machine and the server. If a machine is running multiple **SSH** servers, it may either have multiple host keys or use a single key for multiple servers.

Data Encryption

Encryption, sometimes referred to as privacy, means that your data is protected from disclosure to a would-be attacker "sniffing" or *eavesdropping* on the wire (see the *Threats* section for more details). Ciphers are the mechanism by which Secure Shell encrypts and decrypts data being sent over the wire. A block cipher is the most common form of symmetric key algorithms (e.g. DES, 3DES, Blowfish, AES, and Two fish).

Data Integrity

Data integrity guarantees that data sent from one end of a transaction arrives unaltered at the other

IJCSMS International Journal of Computer Science & Management Studies, Vol. 11, Issue 02, Aug 2011
ISSN (Online): 2231 –5268
www.ijcsms.com

end. Even with Secure Shell encryption, the data being sent over the network could still be vulnerable to someone inserting unwanted data into the data stream.

II. FEATURES OF SSH

The SSH protocol provides the following safeguards:

- After an initial connection, the client can verify that it is connecting to the same server it had connected to previously.
- The client transmits its authentication information to the server using strong, 128-bit encryption.
- All data sent and received during a session is transferred using 128-bit encryption, making intercepted transmissions extremely difficult to decrypt and read.
- The client can forward X11 applications from the server. This technique, called *X11 forwarding*, provides a secure means to use graphical applications over a network.

Because the SSH protocol encrypts everything it sends and receives, it can be used to secure otherwise insecure protocols. Using a technique called *port forwarding*, an SSH server can become a conduit to securing otherwise insecure protocols, like POP, and increasing overall system and data security.

IV. WHY USE SSH?

Nefarious computer users have a variety of tools at their disposal enabling them to disrupt, intercept, and re-route network traffic in an effort to gain access to a system. In general terms, these threats can be categorized as follows:

- *Interception of communication between two systems* — In this scenario, the attacker can be somewhere on the network between the communicating entities, copying any information passed between them. The attacker may intercept and keep the information, or alter the information and send it on to the intended recipient.

  This attack can be mounted through the use of a packet sniffer — a common network utility.

- *Impersonation of a particular host* — Using this strategy, an attacker's system is configured to pose as the intended recipient of a transmission. If this strategy works, the user's system remains unaware that it is communicating with the wrong host.

  This attack can be mounted through techniques known as DNS poisoning or IP spoofing.

Both techniques intercept potentially sensitive information and, if the interception is made for hostile reasons, the results can be disastrous.
If SSH is used for remote shell login and file copying, these security threats can be greatly diminished. This is because the SSH client and server use digital signatures to verify their identity. Additionally, all communication between the client and server systems is encrypted. Attempts to spoof the identity of either side of a communication does not work, since each packet is encrypted using a key known only by the local and remote systems.

III. FUNCTIONALITY OF SECURE SHELL

Secure Shell provides three main capabilities, which open the door for many creative secure solutions.

- Secure command-shell

- Port forwarding

- Secure file transfer

Secure Command Shell

Command shells such as those available in Linux, Unix, Windows, or the familiar DOS prompt provide the ability to execute programs and other commands, usually with character output. A secure command-shell or remote logon allows you to edit files, view the contents of

IJCSMS International Journal of Computer Science & Management Studies, Vol. 11, Issue 02, Aug 2011
ISSN (Online): 2231 –5268
www.ijcsms.com

directories and access custom database applications.

Port forwarding

Port forwarding is a powerful tool that can provide security to TCP/IP applications including e-mail, sales and customer contact databases, and in-house applications. Port forwarding, sometimes referred to as tunneling, allows data from normally unsecured TCP/IP applications to be secured. After port forwarding has been set up, Secure Shell reroutes traffic from a program (usually a client) and sends it across the encrypted tunnel ,then delivers it to a program on the other side (usually a server).

Secure File Transfer

Secure File Transfer Protocol (SFTP) is a subsystem of the Secure Shell protocol. In essence, it is a separate protocol layered over the Secure Shell protocol to handle file transfers. SFTP has several advantages over non-secure FTP. First, SFTP encrypts both the username/password and the data being transferred. Second, it uses the same port as the Secure Shell server, eliminating the need to open another port on the firewall or router. Using SFTP also avoids the network address translation (NAT) issues that can often be a problem with regular FTP. One valuable use of SFTP is to create a secure extranet or fortify a server or servers outside the firewall accessible by remote personnel and/or partners (sometimes referred to as a DMZ or secure extranet) two sides to be unable to communicate with each other.

IV. FILE TRANSFER PROTOCOLS USING SSH

There are multiple mechanisms for transferring files using the Secure Shell protocols.

- Secure copy (SCP), which evolved from RCP protocol over SSH
- rsync, intended to be more efficient than SCP
- SSH File Transfer Protocol (SFTP), a secure alternative to FTP (not to be confused with FTP over SSH)
- Files transferred over shell protocol (a.k.a. FISH), released in 1998, which evolved from Unix shell commands over SSH

V. PROBLEMS WITH SSH PROTOCOL

SSH is not widely supported when compared to the traditional remote access programs. Consequently, mobile users who do not have access to SSH must either revert to the traditional insecure methods or forfeit connectivity. Using security terminology, this lack of access can be seen as a problem in availability. If the insecure methods are used, security is compromised and all the benefits of SSH are lost .

In user authentication, SSH provides backwards compatibility with r*-based programs by supporting .rhosts and /etc/hosts.equiv configuration files. Providing this feature encourages the use of traditional insecure means of connection. Naturally, systems which remain configured in this manner are at risk of traditional r*-based attacks. Kerberos is also supported as a method for user authentication although this system is known to have its own set of security problems.

In remote host authentication, SSH1 uses the RSA public/private key method. The default configuration permits users to accept new public keys of remote hosts without authentication through certificates. Unfortunately, users who choose to accept these public keys are vulnerable to man-in-the-middle attacks. To prevent such an attack, system administrators are responsible for managing the public keys of commonly used hosts. SSH2 addresses this shortcoming by optionally supporting various certificate formats.

Similar problems are present in systems that employ dumb-terminals and Xterminals on a LAN. On these terminals, all processing occurs on other computers located across the network so the flow of unencrypted data (especially passwords) can be intercepted. Therefore, SSH is rendered insecure on these terminals.

User errors can lead to security breaches because they may not be aware that security is compromised if an insecure channel is

IJCSMS International Journal of Computer Science & Management Studies, Vol. 11, Issue 02, Aug 2011
ISSN (Online): 2231 –5268
www.ijcsms.com

traversed anywhere along the communication path. For instance, a user who first telnets to a computer located on the LAN before using SSH to access a remote host will allow hackers to monitor the insecure portion of the path. Such an error is very easy to overlook by the average user and cannot be detected and prevented by SSH.

SSH relies on configuration and key files to determine access rights. Systems that use Sun Microsystems' Network File System (NFS) to access these files pose a major security risk. Since the NFS specification is widely available and packets are transmitted across the local area network (LAN) in clear text, hackers can easily employ NFS sniffers to obtain secret keys, alter public keys, and add public keys.

Since there are numerous security breaches reported and numerous patches issued for SSH [3], [5], [12], [13], system administrators have the tedious task of updating and verifying the security of their system. Due to human nature, system administrators may fail to follow this rapid pace of change. Ignorance may lead to situations similar to the buffer overflow problem where some systems remain unpatched long after a patch has been issued. The original SSH implementation and subsequent patches must be obtained using a secure channel. These packages must be signed by a reputable authority since there is the possibility of obtaining corrupt software. Once a patch is installed, system administrators face the difficult task of verifying that a breach did not occurred before the installation.

To the horror of system administrators, SSH allows tunnelling which can be used to subvert firewalls and breach security policies. It creates a large hole in the firewall that can lead to security breaches in a surprisingly different manner. Hackers can target SSH as a means of penetrating firewalls and attacking internal computers.

VI. PROPOSED SOLUTIONS

All traditional remote access programs, which includes the corresponding daemons and clients, should be removed from the system. Such action will prevent most attempts to use insecure means of communication. Although it may be adequate to remove only the server components (daemons), removing the client components will prevent security breaches on other remote systems.

Strict public host key checking must be enforced. This option is specifiable by the system administrator. New host public keys should never be accepted at face value. If SSH1 is used, connections that present new host public keys should be disallowed unless they can be verified over a secure channel such as through telephone or courier mail. If SSH2 is used, new public host keys should be verified using OpenPGP, X.509, or SPKI certificates. For mobile users of SSH1, public keys of the local system should be stored on a write-protected floppy disk. When away from the local system, the public key can be supplied from the write protected floppy. Users must still trust the system they are using to access the network. With SSH2, the use of certificates also requires a policy for checking certificate revocation lists.

Since the use of NFS is possible, configuration files and key files should be stored and retrieved in an encrypted form. Currently, only the user private key entry is stored in encrypted form in the best scenario. Even with this precaution, the private key file is prone to integrity attacks since only the individual entry is encrypted. The most secure solution involves encrypting and signing all files to ensure confidentiality, integrity, and authenticity while traversing an insecure LAN via NFS. Unfortunately, this solution cannot be implemented by the system administrator alone since it requires alterations to the SSH protocol to ensure complete security.

SSH should not be allowed on dumb-terminals or X-terminals unless communication to the corresponding compute servers is encrypted. Such a policy may create an inequality gap between workstation and terminal users.

Education must be provided to prevent users from introducing an insecure channel along the communication path to the remote host. It is nearly impossible for SSH to detect whether all segments of the communication path are insecure since SSH may be used on only a portion of the path. Such as task lies outside the jurisdiction of SSH and may also lie outside the jurisdiction of the local system. For instance, while a mobile employee is on a business trip, he /she initially

IJCSMS International Journal of Computer Science & Management Studies, Vol. 11, Issue 02, Aug 2011
ISSN (Online): 2231 –5268
www.ijcsms.com

telnets to a gateway and then uses SSH to access the company network.

To restrict tunneling, the SSH protocol must be altered to enable monitoring of tunnel entry and exit points. Monitoring would allow policy Enforcement, denying certain ports from being tunneled in or out of the LAN. Since this option is currently not supported, tunneling remains a serious security risk. The only option remaining is for system administrators to configure SSH with tunneling disabled, which may be too restrictive where access to X11 is required.

All user private keys should be stored in encrypted form to minimize damage caused by breaches in host security. This option is available in SSH but is not mandatory. The SSH implementation must be altered to enforce this restriction. Under these precautions, a hacker who has gained access to a regular user account would be unable to read the user's private key. The passphrase, which is chosen by the user to encrypt his / her private key, should be checked for adequate strength. As well, the security policy should specify that passphrases must never be stored on any medium other than in the user's head.

Both local and remote hosts must be trusted in order to use SSH. Under SSH1, the local system must possess the authentic public key of the remote system. Even under SSH2, where certificates are used to authenticate remote host public keys, the local system must be trusted to contain the genuine public key of the CA or the trusted PGP key. Unfortunately, these judgments cannot be made by system administrators and are left in the hands of users. For instance, mobile employees must determine whether a host can be trusted before using its SSH facilities to access the corporate network.

VII. CONCLUSION

System administrators should adhere to the following guidelines in determining whether SSH will improve security on their system. SSH cannot improve security on systems that contain dumb-terminals or X-terminals connected to the LAN. Any usage from these terminals will create an insecure segment along the communication path. SSH cannot improve security on systems that make use of NFS. SSH cannot improve security if the public keys of all commonly used hosts cannot be authenticated.

Users should adhere to the following guidelines in determining when usageof SSH is appropriate. If a public host key cannot be proven to be authentic, SSH should not be used to communicate with the corresponding remote host. SSH should not be used if the local or remote host makes use of NFS. SSH should not be used if traditional remote access methods are used anywhere along the communication path. Finally, SSH should not be used if the user does not trust the local host or remote host. If usage of SSH is deemed inappropriate, access to the remote system is not possible and users should not revert to the traditional insecure methods.

From the above restrictions, the current SSH specification has only limited real-world applicability. The major barriers are public host key authentication and NFS restrictions. Authenticating all public host keys is currently impractical since most systems use the older SSH1 standard. Since NFS is implemented on most systems, the final set of applicable systems is fairly small.

Even if the problems presented in this paper are resolved, it is only a matter of time before hackers discover new vulnerabilities. SSH must continue improving and system administrators must treat this critical service seriously by keeping their systems updated. Security is a race between hackers and system administrators. Therefore, evaluating the security of a solution involves determining how far one party is ahead of the other.

REFERENCES

[1] E.G. Amoroso, "Fundamentals of Computer Security Technology," Prentice Hall PTR, Upper Saddle River, New Jersey, 1994.

[2] M. Abadi, "Explicit Communication Revisited: Two New Attacks on Authentication Protocols", IEEE Transactions on Software Engineering, vol. 23, no. 3, pp. 185-186, Mar. 1997.

[3] J. Barlow, "SSH Patch Repository," Feb 11, 1999. http://www.ncsa.uiuc.edu/General/CC/ssh/patch_repository/

[4] A. Carasik and S. Acheson, "The Secure Shell (SSH) Frequently Asked Questions," rev. 1.1, Nov. 2, 1999. http://www.employees.org/~satch/ssh/faq/

IJCSMS International Journal of Computer Science & Management Studies, Vol. 11, Issue 02, Aug 2011
ISSN (Online): **2231 –5268**
www.ijcsms.com

[5] "CERT Advisory CA-98.03," Secure Networks Inc, Mar. 2, 1998. ftp://info.cert.org/pub/cert_advisories/CA-98.03.ssh-agent

[6] "Curing remote-access security ailments. ssh, the secure shell, can create a moderately secure network connection," SunWorld, Jan. 1996. http://www.sunworld.com/swol-01-1996/swol-01-sysadmin.html

[7] A. Engelfriet, "The comp.security.pgp FAQ," ver. 1.5, Oct. 22, 1998. http://www.pgp.net/pgpnet/pgp-faq/

[8] P. Galvin, "Enter the secure shell. Turn remote login from security hole to security strength with ssh," SunWorld, Feb. 1998. http://www.sunworld.com/sunworldonline/swol-02-1998/swol-02-security.html

[9] D. Harris, "Diffie-Hellman Key Exchange," Aug. 31, 1998.http://spectral.mscs.mu.edu/NetworksClass/DHKeyExch.html

[10] P. Metzger (chair), "Secure Shell Charter," The Internet Engineering Task Force, Jun. 4, 1999. http://www.ietf.org/html.charters/secsh-charter.html

[11] T. O'Boyle, J. Sergent, and J.S. May, "The Secure Shell," Mar. 6, 1998. http://csociety.ecn.purdue.edu/~sigos/projects/ssh/

[12] A. Polyakov, "SSH and beyond," Aug 14, 1998. http://fy.chalmers.se/~appro/ssh_beyond.html

[13] SSH Communications Security Web Page, Dec. 5, 1999. http://www.ssh.net/

[14] K. Suominen, "Getting Started With SSH," ver. 1.5, Feb. 24, 1998. http://www.tac.nyc.ny.us/~kim/ssh/