Global Information Assurance Certification Paper

Shawn Lewis
Version 1

A Discussion of SSH Secure Shell

The purpose of this paper is to build on the Introduction to SSH Secure Shell paper written by Damian Zwamborn (www.sans.org/infosecFAQ/encryption.intro_SSH.htm). This paper will focus on the following topics of SSH Secure Shell and it's installation on several systems.

Overview of SSH Secure Shell
SSH Secure Shell is a Unix based command interface and protocol used to protect data in transmission between devices. It is commonly used for controlling web, application servers and network appliances remotely. SSH Secure Shell was first created in 1995 by Tatu Ylonen with the release of version 1.0 of SSH Secure Shell and the Internet Draft "The SSH Secure Shell Remote Login Protocol" (www.employees.org/~satch.ssh.faq.ssh-faq.html). It is comprised of a suite of three utilities, slogin, ssh and scp. These utilities are based on earlier versions of UNIX utilities rlogin, rsh and rcp.

The use of these SSH Secure Shell utilities both ends of the connection are authenticated using a digital certificate. Unlike utilities such as Telnet the passwords are encrypted. SSH Secure Shell uses RSA public key cryptography for connection and authentication. These encryption algorithms include DES, 3DES, Blowfish and IDEA (Loshin, Pete).

How SSH Secure Shell works
SSH Secure Shell is a protocol that consists of three mechanisms, the transport layer, user authentication and the connection protocol (www.net.lut.ac.uk). The transport layer runs over TCP/IP of the OSI model. This layer provides the server authentication aspect of the connection. This layer also addresses the confidentiality as well as the integrity of the connection. The second mechanism is the user authentication piece. This will authenticate the client side user to the server. The user authentication will run over the transport layer. The final mechanism is the

connection protocol. This mechanism multiplexes the encrypted tunnel. Transmission of the connection protocol is accomplished over the user authentication layer.

SSH Secure Shell works when the transport layer establishes a connection. The client would send a request after the transport layer has been established. This "connection" would send another service request after the authentication is complete. After these requests have been sent, proper protocols would be agreed upon. The connection protocol mechanism would provide the communication means. X11 connections and TCP/IP ports would be provided for the Secure Socket Shell sessions.

SSH Secure Shell Vendors

Tatu initially offered SSH Secure Shell as a free open source UNIX package. SSH Secure Shell is now currently commercially available on UNIX and non-UNIX platforms. With the commercialization of SSH Secure Shell companies can offer GUI interfaces, documentation, and technical support. There are several vendors that offer SSH Secure Shell packages for sale. One of the most widely regarded is SSH Secure Shell by SSH Communications Security. SSH Communications Security was founded in 1995 and is currently a leader in providing security-related software. Van Dyke Technologies Inc. producer of Secure CRT offers a SSH Secure Shell product that provides a product that uses 56 to 256 bits ciphers for data encryption. Other commercial products offered are Appgate Client and Server by AppGate and F-Secure Tunnel and Terminal by F-Secure. Most vendors offer similar functionality in their products with few vendor specific enhancements or functions.

There are currently many versions and authors of free SSH Secure Shell available on many of the major platforms. UNIX SSH Secure Shell products are the most widely available and developed. OpenSSH suite offers UNIX operators the functions of SSH Secure Shell with a no purchase cost (www.openssh.com). Windows SSH Secure Shell products are quickly becoming readily available for free from many different sources. Simon Tatham's PuTTY is a popular choice with administrators. There are also SSH Secure Shell products available for systems such as the pluggin originally written by Cedric Gourio in 1998 for Java based systems. Macintosh users have the option of downloading MacSSH as well as several others. OS/2, OpenVMS and DOS system administrators also have options if they wish to use SSH Secure Shell.

Why Organizations Should Use SSH Secure Shell

Any organization that transmits data across a network should tighten up security with stronger password protection to secure its data transmission. Such a task can be done through SSH Secure Shell. A SSH Secure Shell program will protect confidential data that flows through the network. A "cracker" could break into a machine by sniffing passwords crossing the wire and exploiting a know vulnerability (Barrett, Silverman). These passwords that are sent, sometimes are transmitted in clear text format. Telnet is an example of a clear text password sent across the network.

SSH Secure Shell is used when remote login to a device or application is necessary.

Administrators have used Unix tools rlogin, rcp (remote copy), rexec (remote execute) and rwho (remote who) to administer their systems. These tools are easy to use and very handy since they are a standard part of the operating system. However these tools are insecure by nature. When using an R utility on a non-Unix box the server could be tricked into giving access to unauthorized parties. These utilities were originally developed to assume that all systems are Unix boxes, thus possibly allowing root access based on a client system. Systems have also been compromised by using the .rhost file. This file contains names of trusted systems and is fairly easy to compromise. With modifying this file a system can be included in the list of trusted systems since the file information is not encrypted.

Use of encrypted communication tools such as SSH Secure Shell, Secure Sockets layer, Secure email, PGP (pretty good privacy) or IP-Layer encryption like IPSec for VPN products can increase network security dramatically. A switched network with VLANs would also decrease the risk. However there are tools available to "sniff" the network. Dsniff is a commonly used tool to "sniff" a switched network. The solution of such malicious activity is to have encryption from end to end.

Installing SSH Secure Shell

The two components of a SSH Secure Shell software package is the server and client. Both components will need to be installed and configured prior to use. The server and the client piece. The server component is not limited to a traditional Microsoft or Novell server but could also be defined as a router, firewall, concentrator or many other devices. These devices however would need to be SSH Secure Shell capable. For example Cisco 7200, 7500 and 12000 series routers are the only supported Cisco router platforms for SSH Secure Shell.

The following sections will cover several examples of basic setups for the server and the client for a SSH system. In the following examples a Cisco 7200 router and a UNIX system will be setup on the server side. A UNIX client will be shown as an example for client installations.

CISCO Router Installation

For the server installation component on a Cisco router two main tasks would need to be completed (www.cisco.com).

- Configure a domain name and host name for the device
- Generate a RSA Key

The below table will perform installation of the SSH Server component for a Cisco 7200 router. The hostname and domain names will need to be setup to identify itself for its clients. Hostnames will be router specific in this explanation. By generating an RSA key this will determine the security for the eventual transmission. There are several configuration settings that could be modified however will not be covered in this document.

Command	Description

Router (config)# **hostname** company	The **hostname** global configuration command is to configure a host name for the router.
Router (config)# **ip domain-name** companydomain	The **ip domain-name** global configuration command is to configure a host domain for the router.
Router (config)# **crypto key generate rsa**	The **crypto key generate rsa** global configuration command is to enable the SSH server for local and remote authentication for the router. Recommended minimum modulus size is 1024 bits. To remove SSH from the server use the crypto key zeroize rsa global configuration command.
Router (config)# **ip ssh {[timeout** 110**]}	[authentication-retries** 4**]**	Optional settings. **Timeout** to specify the SSH negotiation phase. The default is 120 seconds. **Authentication-retries** specify the number of retries. The default is 3.

An example of a 7200 router configuration with a timeout set not to exceed 60 seconds, and no more then 2 authentication retries. The SSH configuration settings are highlighted for example purposes. From a configuration aspect the setup is fairly straightforward with minimal setup and maintenance.

```
version 12.0
no service pad
service timestamps debug datetime msec localtime show-timezone
service timestamps log datetime msec localtime show-timezone
no service password-encryption
service udp-small-servers
service tcp-small-servers

hostname cisco7200

boot buffersize 150000
aaa new-model
aaa authentication login default tacacs+
aaa authentication login aaa7200kw none
enable password enable7200pw

username mcisco password 0 maryspw
username jcisco password 0 johnspw
ip subnet-zero
no ip domain-lookup
ip domain-name cisco.com
ip ssh time-out 60
ip ssh authentication-retries 2

controller E1 2/0
```

```
interface Ethernet1/0
ip address 192.168.110.2 255.255.255.0 secondary
ip address 192.168.109.2 255.255.255.0
no ip directed-broadcast
no ip route-cache
no ip mroute-cache
no keepalive
no cdp enable

interface Ethernet1/1
no ip address
no ip directed-broadcast
no ip route-cache
no ip mroute-cache
shutdown
no cdp enable

no ip classless
ip route 192.168.1.0 255.255.255.0 10.1.10.1

map-list atm
ip 10.1.10.1 atm-vc 7 broadcast
no cdp run

tacacs-server host 192.168.109.216 port 9000
tacacs-server key cisco
radius-server host 192.168.109.216 auth-port 1650 acct-port 1651
radius-server key cisco

line con 0
exec-timeout 0 0
login authentication aaa7200kw
transport input none
line aux 0
line vty 0 4
password enable7200pw

end
```

UNIX-Server Installation

Installation of SSH1 on a UNIX system could be followed per the below instructions if a standard installation of SSH is intended. There are several versions of SSH Secure Shell and flavors of UNIX. Exact file names and locations may be different on each system. Also, directory locations may be specified according to system administrator's preference.

gzcat ssh-1.2.26.tar.gz | tar xf –
cd ssh-1.2.27
./configure –prefix+/usr –without-none –without-rsh
make
make install

By adding "**without-none**" this will enable the option of never to allow an unencrypted password. This will help prevent unencrypted passwords from traveling across a network. The "**without-rsh**" will never allow rshell rhosts as an option. By adding these options into the server configuration, the SSH Secure Shell sessions will be more secure. The "**make install**" switch

will not add the SSH Secure Shell daemon into a system startup file. The system administrator will need to manually add SSH Secure Shell to a system startup file.

Common SSH Secure Shell files on the UNIX server are:

/etc/sshd_config	This file is a server configuration file.
/etc/ssh_config	The client will read this configuration file. Local client configuration files can override the options located in the server's global configuration file.
/etc/ssh_known_host	Host (public) keys that are known are stored within this file.

UNIX-User Installation

The user installation for a UNIX based SSH Secure Shell session is performed on each clients workstation. The user will need to generate a key for the encrypted sessions. The command used is "**ssh-keygen**". This command will launch a random key script that will generate and list the key. The user will be required to save the key. The default location for the key is "$HOME/.ssh/identity". The user will then need to complete the process by entering a passcode and subsequently verifies it. The identification is stored locally in "/u/steve/.ssh/identity" with the public key stored in "/u/steve/.ssh/identity.pub" The client public key is then displayed and the installation is complete.

UNIX-Logging into remote system

For a user to establish a connection to a remote system use either the "slogin" or the "ssh" command. The only parameter that is needed is the name of the remote system. Note that the pass-phrase will not be echoed back to the user.

```
companyx% slogin speters
Enter passphrase for RSA key 'steve@companyx.com': my c@t 83Rt
Last login: Sun May 4 12:04:36 2001 from simdec
[more output from the remote machine]
speters%
```

You can avoid the pass-phrase prompts by keeping the authentication keys in memory. You will only need to type the pass-phrase when you add a key into memory. If your account name on the remote system differs from the one on the local system (the system you are connecting from) you can use the -l switch to specify the remote account name. Creating a configuration file on the host system can alter this.

```
companyx% slogin -l speters companyx.com
Last login: Sun May 4 12:04:36 2001 from companyx.com
[more output from the remote machine]
companyx%
```

Conclusion

While these examples are not complete in the area of SSH Secure Shell systems and implementations, they do show how to establish a secure environment and prevent malicious activity by intercepting private information. As was discussed systems are not only susceptible to non-encrypted passwords but also the overall transmission of company data. These suggestions are not limited to external communication but internal alike. The examples are not by any means a cure all for network data transmission security, but SSH Secure Shell is a good idea to begin with. SSH Secure Shell programs are readily available at no or minimal charge for organizations to investigate if SSH can be used as a encryption or tunneling solution.

References

Barrett Daniel J., Silverman Richard, *SSH, the Secure Shell: Definitive Guide*
February 2001- O'Reilly

Boran, Sean *All About SSH – Part I/II,* May 31, 2001
http://securityportal.com/research/ssh-part-1.html

Loshin, Pete *Sealing the Pipe*, June 2001
Information Security Magazine

Ylonen T., Kivinen T., Saarinen M., Rinne T., Lehtinen *Internet Draft*, May 11, 2000
http://www.net.lut.ac.uk/psst/draft-ietf-secsh-architecture-05.txt

Zwamborn, Damian *An Introduction to SSH Secure Shell,* May 15, 2001
http://www.sans.org/infosecFAQ/encryption.intro_SSH.htm

Secure Shell Version 1 Support
http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120limit/120s/120s5/sshv1.htm

Secure Shell FAQ
http://www.employees.org/~satch/ssh/faq/ssh-faq.html

http://www.vandyke.com/products/securecrt/index.html

http://www.ssh.com/products/ssh/features.html

http://www.openssh.com

http://www.securityfocus.com