SSH Secure Shell

White Paper

Version 1.0, June 2001

SSH Secure Shell

White Paper
Version 1.0, June 2001

SSH Communications Security Oyj

Fredrikinkatu 42
FIN-00100 Helsinki
Finland
Tel: +358 20 500 7030
Fax: +358 20 500 7031

SSH Communications Security Inc.

1076 East Meadow Circle
Palo Alto, CA 94303
USA
Tel: +1 650 251 2700
Fax: +1 650 251 2701

SSH Communications Security K.K.

House Hamamatsu-cho Bldg. 5F
2-7-1 Hamamatsu-cho, Minato-ku
Tokyo 105-0013
JAPAN
Tel: +81 3 3459 6830
Fax: +81 3 3459 6825

http://www.ssh.com
e-mail: ssh-sales@ssh.com

TABLE OF CONTENTS

INTRODUCTION

Today, organizations and their stakeholders are commonly using firewalls and system security measures to protect their private networks. Secure Shell, however, provides a level of security on the network that system security measures and firewalls cannot provide.

The Secure Shell (ssh) technology is considered as the de facto standard for securing remote access connections over IP (Internet Protocol) networks. Secure Shell secures connections over the Internet by encrypting all transmitted confidential data, including passwords, binary files, and administrative commands. The Secure Shell software has made remote management of network hosts over the Internet possible. There are more than two million users of Secure Shell worldwide, including prestigious companies and organizations such as IBM, Sony, Swiss Bank, the US Air Force, NASA, CERN, MIT and Harvard.

SSH Secure Shell has been designed and developed by SSH Communications Security. SSH Secure Shell version 2.x is a powerful, yet easy-to-use application based on the SSH2 protocol. SSH2 is designed to be a complete replacement for the commonly used FTP or Telnet programs, and for rlogin, rsh, and rcp commands.

This following chapters describe the SSH2 technology in general and the SSH Secure Shell version 2.x software in more detail.

SECURITY SOLUTIONS – THE BIG PICTURE

Security solutions can be broken into three different areas: data security, network security, and system security. These areas are defined below. Secure Shell can help to improve security in each of the areas.

Data Security

Data security involves securing the data itself. This can be exemplified in Pretty Good Privacy (PGP), which encrypts email messages and text files. Other examples of data security include disk encryption software and steganography (hiding data of one format into a file of another). Secure Shell does not provide data security; however, Secure Shell can secure a networking port, including the ports used for IMAP and POP3 email transport protocols.

System Security

System security usually involves keeping a computer protected. The data and the network are not a concern in this case. System security can include virus checking, system integrity (for example, Tripwire and MD5 checksums), and Trojan horse and backdoor prevention programs. Secure Shell can help keep your system security intact by allowing a secure point of entrance through the network to your computer.

Network Security

Network security involves protecting any network device. This can include a larger gamut of items than system or data security solutions can. For instance, network security includes security on routers, firewalls, switches, and any computer connected to the network. Another important aspect of network security is to prevent network sniffing. With network sniffing, anyone can reach the information sent on the wire. This includes internal threats, as internal threats are as serious as external attacks.

The security protection offered by Secure Shell is twofold: one, it prevents anyone from sniffing your network traffic; and two, it provides a strong means of authentication to prevent someone from hijacking your session. Another major benefit of Secure Shell is its ease of use, both for the system administrator and the end user.

ENTER SECURE SHELL

Secure Shell secures connections over the Internet by encrypting passwords and other data. Once launched, it transparently provides strong authentication and secure communications over insecure networks.

SSH Communications Security provides Secure Shell products based on the SSH2 protocol, which is designed to be complete replacement for commonly used FTP or Telnet programs, and for rlogin, rsh and rcp commands. SSH Secure Shell for Servers includes all needed components to be able to serve SSH2 clients with defined parameters. Components include Secure Shell 2 daemon (sshd2) and file transfer server (sftp2) amongst others. The most common Linux and Unix environments as well as Windows platforms are supported. SSH Secure Shell 2.x is easy to use. It includes the *ssh2* program, which replaces *rlogin*, *rsh* and *telnet*, the *scp2* program, which replaces *rcp,* and the *sftp2* program, which replaces *ftp*. Complete list of binaries included in SSH Secure Shell 2.4 server product package is shown in Table 1.

Table 1
Binaries included in the SSH Secure Shell Server package

Binary	Replacement to...	Description
ssh2	telnet, rsh	Secure shell client (remote login program)
sshd2	telnetd, rshd	Secure shell daemon
scp2	rcp	Secure copy client
sftp2	ftp	Secure ftp client
ssh-keygen2	-	Authentication key pair generation
ssh-agent2	-	Authentication agent
ssh-add2	-	Adds identities for the authentication agent
ssh-chrootmgr	-	Sets up chroot-ready environment for users

ssh-pubkeymgr	-	Helps to set up public key authentication

SSH Secure Shell Features

Compared to Secure Shell version 1, written by Tatu Ylönen, the founder of SSH Communications Security, SSH Secure Shell version 2.x has been completely rewritten. The results are improved code structure, improved security, and several new features.

The features of SSH Secure Shell 2.x include:

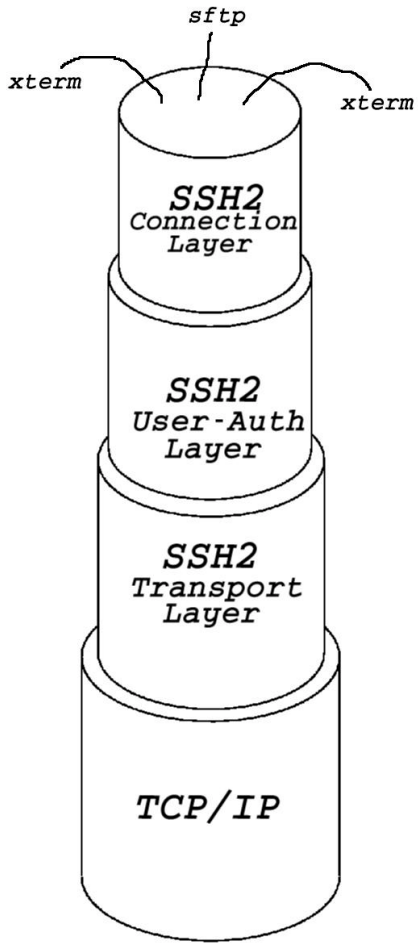
- In addition to command line tools, graphical user interface for *ssh2* and *sftp2* on Windows.
- Graphical configuration tool for *sshd2* on Windows.
- Fully integrated secure file transfer (*sftp*) and file copying. Graphical user interface for *sftp* on Windows.
- Authentication of users, no passwords are sent in cleartext to prevent the stealing of passwords and eavesdropping.
- Authentication of both ends of connection, the server and the client are authenticated to prevent identity spoofing, trojan horses etc.
- Automatic authentication using agents to enable strong authentication to multiple systems with a single sign-on.
- Transparent and automatic tunneling of X11 sessions.
- Tunneling of arbitrary TCP/IP-based applications, such as e-mail.
- Encryption and compression of data for security and speed.
- Built-in SOCKS 4 support.
- Drag-and-drop file transfer (on Windows)
- Several connections can be opened with a single login (on Windows)
- SIA support (on Digital Unix)
- RSA SecurID, PAM, and Kerberos5 support

More information about features can be found in the SSH Secure Shell product documentation.

Secure Shell and the TCP/IP Stack

Secure Shell runs on the application layer of the TCP/IP stack. From here, the SSH2 protocol is defined in several IETF drafts. The Secure Shell protocol is broken into three separate layers: the connection layer, the user authentication layer, and the transport layer. Figure 1 shows how the layers break down on the TCP/IP stack.

Figure 1
Secure Shell on the TCP/IP stack



SSH Secure Shell 2.x uses the SSH2 protocol (Figure 1). SSH2 is "the sequel to the award winning" SSH1 protocol and provides a set of radical improvements to SSH1 (98 % rewritten). It is a protocol for

secure remote login and other secure network services over an insecure network.

The SSH2 protocol consists of three major components:

- Transport layer protocol [SSH-TRANS] provides server authentication, confidentiality, and integrity. It may optionally also provide compression. The transport layer will typically be run over a TCP/IP connection, but might also be used on top of any other reliable data stream.
- User authentication protocol [SSH-USERAUTH] authenticates the client side user to the server. It runs over the transport layer protocol.
- Connection protocol [SSH-CONN] multiplexes the encrypted tunnel into several logical channels. It runs over the user authentication protocol.

The SSH transport layer is a secure low level transport protocol. It provides strong encryption, cryptographic host authentication, and integrity protection. Authentication in this protocol level is host-based; this protocol layer does not perform user authentication. Key exchange method, public-key algorithm (Table 2), symmetric encryption algorithm (Table 3), message authentication algorithm (Table 4), and hash algorithm are all negotiated in this level.

Table 2
Public-key algorithms currently supported in SSH Secure Shell 2.4

Algorithm	Key length (in bits)
DSA	768, 1024, 2048, or 3072
RSA	768, 1024, 2048, or 3072

Table 3
Symmetric encryption algorithms currently supported in SSH Secure Shell 2.4

Algorithm	Key length (in bits)
DES	56
3DES	168
Blowfish	128
Twofish	256
CAST-128	128
Arcfour	128

Table 4
Message authentication (MAC) algorithms currently supported in SSH Secure Shell 2.4

Algorithm
hmac-sha1
hmac-sha1-96
hmac-md5
hmac-md5-96
hmac-ripemd160
hmac-ripemd160-96

The SSH authentication protocol is a general-purpose user authentication protocol (Table 5). It is intended to be run over the SSH transport layer protocol [SSH-TRANS]. This protocol layer assumes that the underlying protocols provide integrity and confidentiality protection.

Table 5
Authentication methods available in SSH Secure Shell 2.4. The administrator can select a single authentication method or a combination of methods.

Method
Unix password
Public key
Host based
PAM (Pluggable Authentication Modules)
SecurID tokens
Kerberos

The SSH Connection Protocol has been designed to run on top of the SSH transport layer and user authentication protocols. It provides channels that can be used for a wide range of purposes. Standard methods are provided for setting up secure interactive shell sessions and for forwarding ("tunneling") arbitrary TCP/IP ports (Figure 2 and Table 6) and X11 connections.

The SSH2 protocol complies with the upcoming *secsh* Internet standard. More information about SSH2 protocol can be found from the SSH2 Internet-Drafts which are available at http://search.ietf.org/ids.by.wg/secsh.html.

- SSH Protocol Architecture: draft-ietf-secsh-architecture-07.txt
- SSH Connection Protocol: draft-ietf-secsh-connect-09.txt
- SSH Transport Layer Protocol: draft-ietf-secsh-transport-09.txt
- SSH Authentication Protocol: draft-ietf-secsh-userauth-09.txt

Secure Shell Protection

Secure Shell is designed to protect trusted connections between two machines from attacks on an external network and an internal network. Secure Shell provides the protection with strong authentication by using public-key algorithms (Diffie-Helman Key Exchange and Digital Signature Algorithm) to prevent spoofing, session hijacking, and man-in-the-middle attacks. Additionally, Secure Shell prevents eavesdropping by encrypting the connection even before the user sends their remote username to the server over the wire.

However, Secure Shell is not designed to protect against flaws inherit to the operating system such as a poorly developed IP stack or insecure storage of passwords. In addition, if someone was to attack and obtain root access on a Unix machine (or administrator access on a Windows NT machine), Secure Shell is also compromised as nothing can defend against a successful root-level attack.

Secure Shell Features

Secure Shell has two prominent features: secure terminal emulation and secure file transfers. In many ways, these Secure Shell functions operate similar to Telnet and FTP, respectively. In addition, Secure Shell supports connection forwarding.

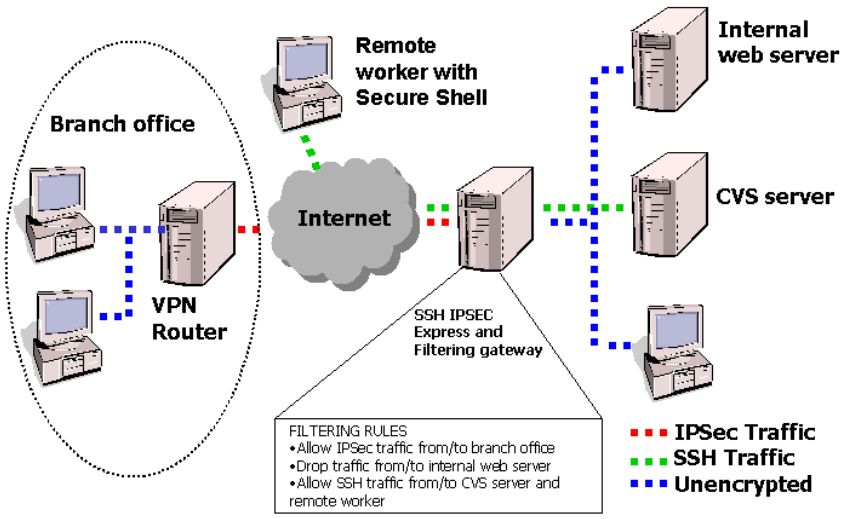
Connection forwarding allows Secure Shell to forward any type of TCP-based connection through the port its server is listening on. This includes standard TCP ports, X11 traffic, and the identity of Secure Shell public keys stored in memory. This enables many system administrators to incorporate virtual private network (VPN) functionality for their remote users when they do not have an immediate need for a full-functionality VPN.

Secure Shell and Firewalls

Secure Shell is not a firewall, nor does it provide the same type of filtering that a firewall does. Where a firewall works similar to a router, Secure Shell works similar to the Berkeley "r" services, Telnet, and File Transfer Protocol (FTP). However, Secure Shell can work quite nicely with firewalls in several different ways.

Secure Shell can provide users a secure remote connection either through the firewall, or for administrators, to the firewall. Many firewalls are installed directly to an operating system (such as Solaris or Linux), and Secure Shell enables the administrator to have a secure remote connection directly to the firewall. Secure Shell can be secured to the level where the administrator is the only user allowed to login.

Figure 2
Secure Shell and the corporate network



For users who would like to connect to machines outside the firewall, Secure Shell provides them privacy and the ability to login to their remote machine without having to worry about someone reading their network traffic. Additionally, the users can copy files to and from a

remote server without having anyone notice what files they are transferring.

SETTING UP SSH SECURE SHELL

Many features of SSH Secure Shell are configurable, depending on your setup. Depending on how secure you want your system to be, you may turn off some of the functionality to further prevent someone from tampering with your system. The table below touches some of the more common features that administrators configure Secure Shell for.

Table 6
Features and their configuration options in SSH Secure Shell 2.4

Feature	Configuration options
User authentication	Password, Pluggable Authentication Modules (PAM), SecurID token, user public key, host-based, Kerberos5
Ciphers	3DES, CAST-128, Arcfour, Blowfish, Twofish, DSA
Hash functions	MD5, SHA-1
Forwarding connections	X11, TCP, and authentication agent (stores identities in memory)
Secure file transfer	Can be configured on some operating systems so that users can only transfer files
Data compression	Improves performance on some slow connections

More information can be found in the SSH Secure Shell product manuals.

USING SECURE SHELL IN YOUR ENVIRONMENT

Depending on the setup of your environment, Secure Shell can provide you with many uses to make your security stronger while being transparent to your users. These include secure remote command execution, forwarding insecure protocols, and advantages for a development environment.

With SSH Secure Shell for Windows Servers, many of the functionalities of Secure Shell that were previously available only for Unix are now available for Windows. This also means that there is an increased interoperability between the Unix and Windows operating systems–this time, that connection is secure.

Remote Command Execution

Secure Shell has the functionality to run commands remotely through a secure connection. This enables system administrators and end users to obtain information about a remote system without having an interactive login session. For example, if the system administrator wants to find out the disk usage of one of the Unix machines he maintains remotely, he would execute the query using Secure Shell.

Taking Advantage of Port Forwarding

Port forwarding enables system administrators to secure otherwise insecure network protocols through their network. This includes email protocols (SMTP, POP3, and IMAP), database connections that use TCP (including Oracle products), X11 applications including X-based applications, Concurrent Versions System (CVS), which is used by developers to centralize their code base, and Windows GUI forwarding through Virtual Network Connection (VNC, offered under the GNU license by AT&T).

Remote Backups with Secure Shell

For many system administrators today, Secure Shell has provided a valuable tool for creating secure remote backups. Secure Shell can either tunnel remote backup applications like Amanda that are TCP-based, or it can use the data dump command (dd) and be piped through Secure Shell.

Programming Advantages with Secure Shell

Secure Shell can also be easily automated on both the Unix and Windows operating systems. For Windows, many users use the GUI to interface with a remote computer. For those wishing to automate Secure Shell, there are command-line interfaces for the Windows client that enables inclusion into batch files and other programs.

For UNIX programmers, it is easy to include the command-line call to ssh2, scp2, or sftp to connect to a remote server. In many cases for Unix, it's easy to set up host-based authentication, which enables two trusted machines to connect without the need for a public key or a system password.

CONCLUSION

SSH Secure Shell is an essential tool to your network security that helps compliment your system and data security. With SSH Secure Shell, network traffic is encrypted and the administrators can decide which means of authentication they require. Additionally, SSH Secure Shell enables you to create secure remote backups and tunnel other TCP-based traffic.