Global Information Assurance Certification Paper

An Introduction to SSH Secure Shell

Damian Zwamborn

GSEC Assignment v1.2d

Purpose

This paper is meant as an introduction to SSH Secure Shell with emphasis on the latest version SSH2. When I was researching this paper, a lot of questions came to mind. I have tried to answer all those questions:

- Background and licensing.
- The SSH1 and SSH2 protocol.
- How does SSH work?
- Why SSH? What are some typical security threats that SSH deters?
- Extended descriptions of SSH features.
- SSH Implementations.
- A few links to popular clients.
- More helpful links.

Overview

SSH Secure Shell is a replacement for the Berkeley 'r' commands, telnet and ftp functionalities. Introduced in the BSD (Berkeley) version of UNIX, the 'r' commands (rsh, rlogin, rcp) allow you to communicate with remote systems. Although they're not as extensive as ftp and telnet, the 'r' commands are useful tools for extending the reach of your network.

SSH functions as a type of tunnel for encoding login procedures. All connections between the local and the remote hosts are encrypted, protecting the data sent between these machines. Secure shell provides several security improvements over the telnet, ftp and rlogin protocols. In particular, passwords are never sent over the network in a clear text format as they are when using telnet, ftp or rlogin. This encryption makes it difficult for someone to breach the confidentiality of your data and/or compromise passwords. SSH Secure Shell is based on the SSH2 protocol that is standardized by IEFT in a draft format (www.ietf.org/ids.by.wg/secsh.html).

Secure Shell does not close all network security holes, but it is one step toward a more secure network.

SSH Secure Shell Background

SSH Secure Shell has been around since 1995 and is widely used in many Unix systems all over the world. SSH was originally an academic project authored and distributed by Tatu Ylonen of the University of Finland. SSH Secure Shell was commercialized in 1998 by SSH Communications Security (www.ssh.com/), which sells implementations for both Windows and UNIX systems.

The SSH1 and SSH2 protocol

As of 1 May 2001, SSH Secure Shell 1.x is no longer available from the SSH Communications Security site. SSH1 has many security flaws and the SSH2 protocol adds additional security, If fact, SSH2 is a complete rewrite of the SSH protocol. SSH Communications Security has stated: "SSH1 (Secure Shell 1) was designed in 1995 … it [has] became apparent that there were inherent security flaws in SSH1. These flaws include a weak hash and susceptible encryption algorithms. These flaws fall in the following categories: access control and authentication, data integrity, confidentiality, and connection redirection" (Ref 1).

A summary of vulnerabilities in the SSH1 protocol can be found at the CERT Coordination Center: www.kb.cert.org/vuls.

For the sake of clarity, this paper refers to the SSH2 protocol unless explicitly stated otherwise.

Licensing

The SSH Communications Security FTP site offers downloadable source code and binaries (UNIX platforms and Windows) for both SSH1 and SSH2. However, the licensing terms allow only SSH1 to be used without restriction. Commercial enterprises must license SSH2 products on a per-seat basis.

Open Source implementation of SSH

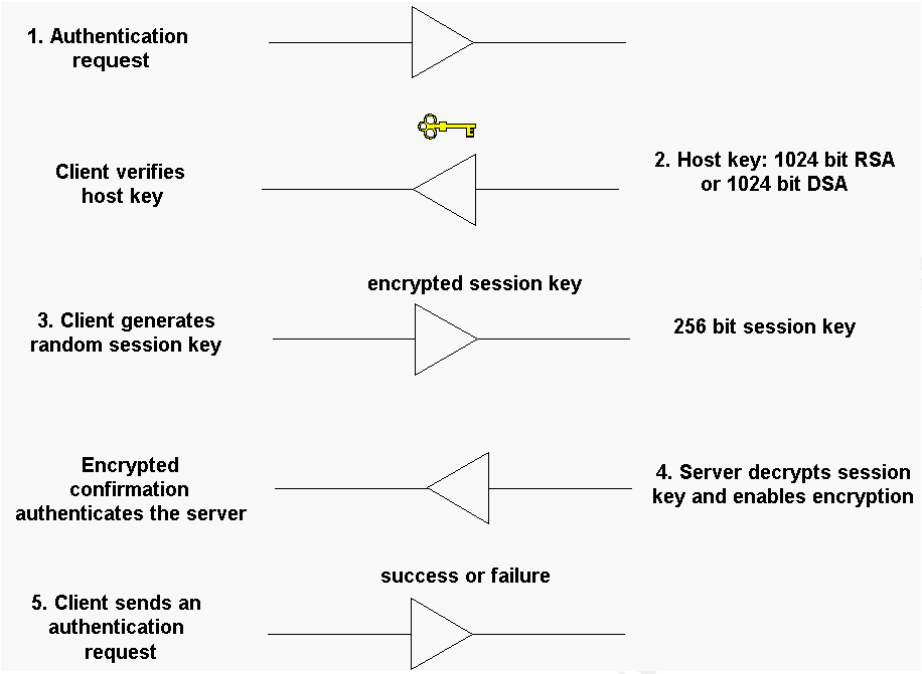
There is also an Open Source implementation of SSH, available from OpenSSH (www.openssh.com/). This implementation supports both the ssh1 and ssh2 protocols and has no restrictions on its use. The Open Source implementation is largely compatible with the commercial version, differing mainly in not providing support for IDEA encryption because of patent restrictions.

The FreeSSH (www.freessh.org/) site maintains up-to-date links on various free and Open Source SSH implementations as well as commercial enterprise implementations.

How does SSH work?

The SSH2 login procedure can be illustrated in the following client/server model.

Note: SSH1 uses server and host keys to authenticate systems where SSH2 only uses host keys. A server key (768 bit) is generated every hour by default and is not saved in a file. The server key ensures the encoded data can no longer be decoded once the server key, after an hour, has been regenerated (in the case that the private host key is ever compromised).



1. The client makes a connection to a server.

2. The server identifies itself with it's public host key . The key length is 1024 bit RSA or DSA. The client looks in its local database to verify the public host key is authentic/known. An unknown key is added to the database or the session can be broken. If the client determines the host key does not belong to the server the client is alerted (SSH generates a warning).

3. The client then generates a random 256 -bit number and chooses an encryption algorithm (eg. 3DES). The random number is then encoded with RSA or DSA. Pure RSA/DSA authentication never trusts anyt hing but the private key. The encoded key is then sent to the server. The host key ensures the authentication of the particular server.

4. The server decodes the RSA/DSA encryption and reconstructs the session key. Furthermore, the server sends the client, v ia the encoded session key, a confirmation. The rest of the session is encrypted using a symmetric cipher.

5. The client then sends a username authentication request. The server replies with a success or failure.

Security Threats

TCP/IP was not designed wi th security in mind. That is why it is important to implement security techniques such as SSH Secure Shell.

A quick review of some valuable terms:

- Packet sniffing: A packet sniffer is a utility that sniffs raw packet data without manipulating the data. P acket sniffers observe, display, and log traffic. A powerful

aspect of packet sniffers is their ability to place the network adapter machine hosting the sniffing software into promiscuous mode. Network adapters running in promiscuous mode receive not only the data directed to the host's machine but also all of the traffic on the physically connected local network. Malicious hackers use packet sniffers gathering information.

- Spoofing: Spoofing is the creation of TCP/IP packets using somebody else's IP address. Routers use the destination IP address in order to forward packets through the Internet, but ignore the source IP address. The destination machine only uses that address when it responds back to the source.

SSH connections are encrypted, protecting the data sent between machines. This encryption makes it difficult for someone to breach the confidentiality of your data and/or compromise your passwords. Typical attacks that SSH protects against are:

- Trojan horses: Trojans are programs that masquerade as normal programs on an unsuspecting host but instead contain code designed to break into a system. SSH authentication happens on both ends of a connection and therefore, for practical purposes, eliminating unauthorized connections.
- DNS spoofing. "DNS spoofing is a term used when a DNS server accepts and uses incorrect information from a host that has no authority giving that information. DNS spoofing is in fact malicious cache poisoning where forged data is placed in the cache of the name servers. Spoofing attacks can cause serious security problems for DNS servers vulnerable to such attacks, for example causing users to be directed to wrong Internet sites or e-mail being routed to non-authorized mail servers" (Ref 2). Follow reference link for detailed explanation. DNS is vulnerable to spoofing because of the absence of authentication.
- Man-in-the-middle attack. A man-in-the-middle attack "is one in which the attacker intercepts messages in a public key exchange and then retransmits them, substituting their own public key for the requested one, so that the two original parties still appear to be communicating with each other directly. The attacker uses a program that appears to be the server to the client and appears to be the client to the server" (www.whatis.com/). A well-known tool for this is dsniff (www.monkey.org/~dugsong/dsniff). Since most authentications only occur at the start of a TCP session, this allows the hacker to gain access to a machine.

  SSH offers the following solution to prevent public key substitution: "SSH2 automatically maintains and checks a database containing public keys of hosts. When logging on to a host for the first time, the host's public key is stored to a file in the user's personal directory. If a host's identification changes, SSH2 issues a warning and disables password"(Ref 3).

  A note regarding SSH1 and man-in-the-middle attacks:

  > The protocol [SSH1] provides the option that the server name-host key association is not checked when connecting the host for the first time. This allows communication without prior communication of host keys or certification. The connection still provides protection against passive listening; however, it becomes vulnerable to active man-in-the-middle attacks. Implementations SHOULD NOT normally allow such connections by default, as they pose a potential security problem. However, as there is

no widely deployed key infrastructure available on the Internet yet, this option makes the protocol much more usable during the transition time until such an infrastructure emerges, while still providing a much higher level of security than that offered by older solutions (e.g. telnet and rlogin) (Ref 4).

Features and Specifications

Authentication methods.

- Host-based authentication: password protection with 1024 bit RSA -key (public-key cryptographic algorithm).
- Ability to add certificate and public key authentication.
- Password authentication.
- Public key algorithm support: DSA and Diffie-Hellman key exchange.
- PGP key support.

Data encryption for confidentiality and integrity.

- Encryption algorithms: DES, 3DES Blowfish, Twofish, Arcfour, CAST128 -CBC, 128 bit AES, or 256 bit AES.
- Hash Algorithms: MD5 and SHA1.

Additional functionality.

- File transfer. Files can be copied remotely using a utility called scp (secure copy).
- SFTP (secure file transfer protocol) was introduced with SSH2. It is independent to the rest of the SSH2 protocol suite (see www.ietf.org/ids.by.wg/secsh.html). SFTP-server is not called directly but through the SSH2 daemon – the ssh2 protocol therefore secures the connection.
- TCP/IP port forwarding: SSH supports port forwarding over a secure tunnel. You configure your SSH client to accept connections on the local machine for certain ports. Any data that is sent to these ports is then forwarded and returned across the tunnel. On the other side of the tunnel, the SSH server passes the data back and forth to a server you wish to access. Typical implementations of port forwarding are services that have no encryption of their own built-in such as FTP, IMAP, SMTP and POP3. Note: only root can redirect privileged ports.
- X11 connections for secure X Window System sessions are a popular implementation. SSH Communications Security offers special support for this feature because it so popular. SSH creates a fake X server (Fake Xauthority information) on the same machine that the SSH client is run. SSH then functions as a go-between between the connection and forwards it to a real X server over a secure connection.
- TCP Wrapper support. The TCP Wrappers is a public-domain tool/package you can monitor and filter incoming requests for the SYSTAT, FINGER, FTP, TELNET, RLOGIN, RSH, EXEC, TFTP, TALK, and other services. It is used to restrict inbound network access to the services defined in the /etc/inetd/inetd.conf file. TCP Wrappers is controlled by two configuration files: /etc/hosts.allow and /etc/hosts.deny. If they do not exist or are empty, TCP Wrappers will allow all hosts to access all services.
- Built-in SOCKS 4 and 5 support for traversing firewalls.

- Support for SSH1 fallback functionality. The SSH1 and SSH2 protocols are not compatible. SSH1 clients cannot connect to a SSH2 daemon. The SSH2 daemon can be configured to start up a SSH1 daemon for a SSH1 client. (Must be installed with compatible option). The SSH1 and SSH2 daemon can be running on two separate machines or on the same machine.
- Multiple channel support. "All terminal sessions, forwarded connections, etc. are channels. Either side may open a channel. Multiple channels are multiplexed into a single connection" (Ref 5).
- Distributed key management. The advantage of distributed key management is that there is no hierarchy. This avoids having a single target for attacks as every machine can hold it's own keys. It is also possible to configure these machines to change session keys every hour.

SSH Implementations .

The SSH2 protocol can be implemented in various ways. In addition to encrypted terminal connections the SSH protocol can be implemented, for example, as a complete VPN solution. At the risk of going beyond the scope of this paper, I recommend going to www.ssh.com/products , and www.f-secure.com/products , to get an idea of popular implementations/solutions.

Popular Clients

SSH Communications Security. URL: www.ssh.com .
F-Secure. URL: www.f-secure.com . In the past, SSH Communications Security did not sell Secure Shell but were licensing it through F-Secure. F-Secure still licenses the UNIX server but both companies now develop their own clients.
Van Dyke Technologies. URL: www.vandyke.com .
Open SSH. URL: www.openssh.com .
For a complete list of free and licensed clients, visit the www.freessh.org website.

References (Ref):

1. SSH Communications Security. " SSH1 vulnerabilities ". http://www.ssh.com/products/ssh/cert/vulnerability .html (12 May, 2001).
2. MEN&MICE. "What is DNS Spoofing?" IWS - The Information Warfare Site. URL: http://www.iwar.org.uk/comsec/resources/dns/DNS_spoofing.htm (12 May 2001).
3. SSH Communications Security. "SSH2 Functionality". SSH Secure Shell for Workstations Windows Client: User Manual. URL: http://www.ssh.com/products/ssh/winhelp21/8_1_SSH_2_Functionality.html (12 May 2001).
4. Network Working Group. "SSH Protocol Architecture". Internet-Drafts IETF. URL: http://www.ssh.com/tech/archive/secsh/architecture.txt (12 May, 2001).
5. Network Working Group. "SSH Connection Protocol". Internet-Drafts IETF. URL: http://www.ssh.com/tech/archive/secsh/connect.txt (12 May, 2001).

Extra references:

- RSA Security. URL : www.rsa.com .

- The Official Secure Shell FAQ. URL: www.employees.org/~satch/ssh/faq .
- Indiana University security resource page.
  URL: www.uwsgiu.edu/security/ssh.html .
- Colorado School of Mines Secure Shell resource page.
  URL: www.mines.edu/Academic/computer/security/tools/ssh/#ab_out.
- Crawford, Chuck. " SSH, Secure Shell ". Nov 3 2000.
  URL: www.sans.org/infosecFAQ/authentic/SSH.htm .

Additional Security Alerts of Interest

- Openwall advisories. "Passive Analysis of SSH (Secure Shell) Traffic. Determining possible password lengths". Mar 19, 2001.
  URL: www.openwall.com/advisories/OW-003-ssh-traffic-analysis.txt.
- Russel, Christopher R. "Penetration Testing with dsniff" . Feb 18, 2001. URL: www.sans.org/infosecFAQ/threats/dsniff.htm .
- CORE SDI S.A. Security Advisory. "SSH protocol 1.5 session key recovery vulnerability". Feb 7th, 2001.
  URL: www.core-sdi.com/advisories/ssh1_sessionkey_recovery.htm .
- Arce, Iván. "SSH1 CRC-32 compensation attack detector vulnerability". Feb 08 2001. URL: www.securityfocus.com/archive/1/161448 .