Global Information Assurance Certification Paper

Compiling OpenSSH with S/key and TCP Wrappers on Solaris

Robert Oxonian

GSEC Version 1.4b Option 1

Abstract

OpenSSH is a free open source adaptation of the SSH Secure Shell, which is a widely utilized tool for starting a remote shell securely. The advantages of building the OpenSSH software package in place of using pre-built binaries are discussed, followed by the benefits of compiling OpenSSH with S/key and TCP Wrappers. The concepts of OpenSSH, S/key and TCP Wrapper support are explored. Finally, instructions on a custom configuration, build and install of OpenSSH with S/key and TCP Wrapper support on Solaris is described.

1 Introduction

Why Build OpenSSH with S/key and TCP Wrappers?

The OpenSSH software suite is excellent, however, the possibility of "hack attacks" against OpenSSH can not be entirely discounted. No software, unfortunately, can be considered completely full proof from a security standpoint. New holes in security software are always being discovered and will continue to be found as attackers get more skilled.

Multiple layers of protection should be implemented as a "best practice", which is also known in the computing security field as "defense in depth". Practicing defense in depth creates an additional security mechanism for an attacker to defeat should the first line of defense fail.

The compiling and installation of OpenSSH with support for TCP Wrappers and S/key can make it stronger security software when the wrappers and S/key are enabled. Extra layers of protection are in place because the wrappers will allow only privileged hosts access to the SSH service. If an attacker somehow successfully completes a "man in the middle attack" on SSH [1], the usage of S/key would render the "sniffed" password useless because it will only be valid for one authentication.

Pre-built binaries may or may not have desired features or options enabled by default, and therefore are not as flexible as user-built software. Building software oneself eliminates being restricted to features that are present, nonexistent or unsatisfactory.

The next section is a brief overview of OpenSSH, S/key One Time Passwords and TCP Wrappers. Section 3 describes the software requirements for compiling and installing S/key and TCP Wrappers with OpenSSH. Setup and getting started with S/key is described in Section 4. Configuration and installation of TCP

Wrappers and OpenSSH on Solaris are discussed in Sections 5 and 6 respectively. A brief discussion in Section 7 will conclude this paper.

2 Overview of OpenSSH, S/key and TCP Wrappers

OpenSSH

OpenSSH is a free implementation of SSH, which is widely used as a secure replacement for telnet, rlogin, and rsh [2]. Programs such as telnet and ftp send data including passwords over the network in "clear text", and are therefore susceptible to "sniffing" attacks. OpenSSH encrypts all traffic and greatly inhibits the ability for a malicious user to perform sniffing, session hijacking, and other network attacks [2]. "Privilege Separation" makes it difficult to run exploits for error in code such as buffer overflows because the unprivileged child process is the only process that handles network data. An excellent description is discussed in the "Privilege Separated OpenSSH" document [3].

S/key

S/key is a one-time password authentication system that is secure from passive attacks against reusable passwords [4] [5]. Systems that use the "login" program and other applications requiring authentication can utilize S/key. Sequences of passwords are generated from an initial pass-phrase, and are good for only one login or authentication. In addition to protection against "sniffing", a useful mechanism of S/key is the tracking of the next expected valid password sequence number. An unexpected sequence number displayed at the login prompt can be evidence that someone other than the user somehow authenticated with his or her username.

TCP Wrappers

TCP Wrappers provide the ability to monitor and filter incoming requests for a wide range of network services such as finger, ftp, telnet, rlogin, and rshd [6] and was originally written by Wietse Venema [7] [8]. The inetd daemon listens for incoming requests for services such as telnet, runs the requested program, and then resumes waiting for further connections. The TCP Wrapper package provides a program called "tcpd" that will perform a series of additional checks instead of directly invoking the requested network service immediately. After the appropriate checks are made, the wrapper will execute the requested server program and then go away [6].The most common use for TCP Wrappers is access control, permitting only hosts explicitly specified in the "hosts.allow" file to connect to the service and deny hosts listed in the "hosts.deny" file.

3 Requirements

Please note this installation is tested on Solaris 8 and 9, and may or may not work with previous versions.

Source code packages for OpenSSH, S/key, TCP Wrappers IPv6, Zlib and OpenSSL are necessary before implementation can begin. OpenSSL requires Perl 5 and may need to be acquired if not present on the system. Optional software includes the Pseudo Random Number Generator Daemon (PRNGD) or the Entropy Gathering Daemon (EGD) for systems prior to Solaris 9 that do not support /dev/random [15]. The homepages for the aforementioned software where sources can be found are:

OpenSSH v3.5p1 [2]:
http://www.openssh.org

S/key v1.1.5 [9]:
http://www.sparc.spb.su/solaris/skey

TCP Wrappers IPv6 v7.6 by Casper Dik [8]:
ftp://ftp.porcupine.org/pub/security/index.html#software

Zlib 1.1.4 [10]:
http://www.gzip.org/zlib

OpenSSL 0.9.6 or greater [11]:
http://www.openssl.org

Optional software:
Perl 5 [12]:
http://www.perl.org

PRNGD [13]:
http://www.aet.tu-cottbus.de/personen/jaenicke/postfix_tls/prngd.html

EGD [14]:
http://egd.sourceforge.net

Installation for Zlib, OpenSSL and optional software will not be covered. Please refer to the instructions included with the software.

4 Setting Up and Getting Started with S/key

Building S/key is fairly straightforward and has similar installation instructions as most software packages. The directions included with the source are straightforward and should be referred to for installation.

Please note that the following instructions are referring to the S/key port mentioned in section three. The aforementioned S/key port is the only known version supported by OpenSSH [15]. The following instructions may be similar to other versions of S/key, but may have subtle differences.

Once installation of S/key is complete, the system must be initialized so that users can use one-time passwords. As the root user type the command:

 # skeyinit

to initialize S/key one-time passwords for root. To initialize S/key for another user simply type at the command prompt:

 # skeyinit username

"Skeyinit" should only be run from console or over a secure encrypted network session. If a secure channel is unavailable, "skeyinit" must be executed with the –s option. Please refer to the "skeyinit" man page for details of initializing passwords over an insecure network [16].

The message "Enter secret password:" will appear requesting a pass-phrase with a minimum of ten characters in length. The specified pass-phrase is used to generate the S/key one-time passwords, and also to retrieve passwords later. By default a sequence of 99 passwords are generated with an md5 hash algorithm. Changing the default number of password sequences generated and hash algorithm type is documented in the "skeyinit" man page [16].

The output of "skeyinit" provides details of the generated keys. An example of output from "skeyinit" is shown below:

```
ID root skey is otp-md5 99 deva90402
Next login password: FACE QUIT REB THUG IDLE LYE
```

The "otp-md5" on the first line is the algorithm identifier, in this case an md5 hash algorithm. The "99" denotes the sequence integer, specifying that root has a set of 99 passwords sequences. The last field on the first line, "deva90402", is the seed [5], or sometimes described as the key.

To output a list of password sequences for use, the following command may be used:

```
# skey -n 5 99 deva90402
```

The above command will list five passwords starting with the sequence number 99 for the user key “deva90402” after the pass-phrase is entered. When the “-n” option is omitted, the number of passwords displayed will default to one[4]. An incorrect set of passwords will be displayed if the wrong pass-phrase is entered. Example output from the above command is:

```
# skey -n 5 99 deva90402
Reminder - Do not use this program while logged in via telnet or rlogin.
Enter secret password:
95: BALM HAW TOO DIG GALT FLOG
96: ARAB EARL HAW HAT OUST HUSH
97: HIDE WAST CAL MUCH LIEU BENT
98: SHAW FILE CHOU BEG THEY DEAD
99: FACE QUIT REB THUG IDLE LYE
```

S/key is now available for use on this system for user root and other users initialized by “skeyinit”.

5 Configuring and Installing TCP Wrappers

The modified version of standard TCP Wrappers version 7.6 with IPv6 support is necessary for compatibility with the implementation of IPv6 [17] first seen in Solaris 8. The older standard version of TCP Wrappers will not function properly.

The TCP Wrapper software package does not come with a configure script thus the “Makefile” must be modified manually.

Make the following changes to the TCP Wrapper “Makefile”:

```
# SysV.4 Solaris 2.x OSF AIX
REAL_DAEMON_DIR=/usr/sbin

# System dependencies: selection of non-default object libraries
LIBS   = -lsocket -lnsl              # SysV.4 Solaris 2.x

# System dependencies: working around system bugs.
# BUGS = -DGETPEERNAME_BUG -DBROKEN_FGETS - \
  DLIBC_CALLS_STRTOK
```

Comment out the above workaround for older system bugs. Please see the comments in the “Makefile” [18] for details.

```
# System dependencies: whether or not your system has IPV6
```

```
IPV6 = -DHAVE_IPV6
```

The following changes to the TCP Wrapper Makefile are optional but recommended.

```
# Optional: Turning on language extensions
STYLE = -DPROCESS_OPTIONS     # Enable language extensions.
FACILITY=LOG_LOCAL7
```

Enabling language extensions allows the usage of keywords such as “allow”, “deny” and “banners”. “FACILITY” defines to which syslog priority successful connections are logged [18]. “LOCAL7” is one of the user definable syslog facilities and is used in place of the default facility “LOG_MAIL”. “LOCAL7” is used as a preference to keep wrapper logs separate. For example, an entry in “syslog.conf” can be created as follows:

```
local7.info                                    /var/adm/wrapper.info
```

At least one tab or space may be necessary in between the selector field local7.info and the action field /var/adm/wrapper.info. For further details on configuring syslog, please refer to the “syslog.conf” man page [19].

```
# Optional: Reduce DNS load
DOT= -DAPPEND_DOT
```

The APPEND_DOT feature stops the waste of cycles in DNS behavior of first appending substrings of the server domain [18].

If the default pathnames to the access control tables are undesirable, the following definition may be changed:

```
# Optional: Changing the access control table pathnames
TABLES = -DHOSTS_DENY=\"/etc/hosts.deny\" - \
DHOSTS_ALLOW=\"/etc/hosts.allow\"
```

Please note that the backslash ‘\’ at the end of the line containing “TABLES” denotes the UNIX flag to continue on the next line. The above option should be on the same line.

After modifications to the wrapper makefile are complete, the command

```
make sunos5
```

will build the wrapper programs and libraries. The wrapper program “tcpd” should be copied to the location specified by “REAL_DAEMON_DIR” earlier. The library “libwrap.a” and the header file “tcpd.h” should be placed in convenient directory

paths for later use by other programs where wrapper support is desired. Suggested locations for “libwrap.a” are /usr/local/lib or /opt/local/lib. The directories /usr/local/include or /opt/local/include may be favorable for “tcpd.h”.

To wrap programs such as telnet, the “inetd.conf” file will need to be modified. For example to wrap telnet the entry in “inetd.conf” would appear as follows:

```
telnet  stream  tcp    nowait  root    /usr/sbin/tcpd  in.telnetd
```

The above instructs ”inetd” to execute “tcpd” first instead of /usr/bin/in.telnetd immediately. Those who are unfamiliar with “inetd” should read the “inetd.conf” man page for details [20].

The final step in configuring TCP Wrappers is to set up the access control table files “hosts.allow” and “host.deny”. Access will be granted when a daemon and client pair match, and is denied when a similar type pair is matched in “hosts.deny”. Below is an example of access control tables:

hosts.allow:

```
in.telnetd: 192.168.1.0/255.255.255.0: ALLOW
```

hosts.deny:

```
ALL : ALL@ALL : DENY
```

The above access control tables will allow all clients coming from the 192.168.1.x network to connect to the telnet service on to the server, and deny everything else. For details on the format of the access control tables, please consult the hosts_options(5) [21] man page.

6 Configuration and Installation of OpenSSH

After installing the software requirements, compiling and installing OpenSSH is fairly straightforward.

The first step before compiling OpenSSH is to set the appropriate options to the “configure” script, which is included with most software packages. The “configure” script queries the system and sets configuration parameters appropriately. At the command prompt, execute the “configure” script in the following manner:

```
./configure --prefix=/opt/local \
            --with-skeys=/opt/local \
            --with-tcp-wrappers=/opt/local/lib \
            --with-ssl-dir=/opt/local/openssl \
            --with-privsep-user=privsep-user \
            --with-md5-passwords
```

The “--prefix” option specifies to install OpenSSH in a top directory of /opt/local instead of the default /usr/local. The “--with-skeys”, “--with-tcp-wrappers” and “--with-ssl-dir” options specify the directory paths to the corresponding libraries. The “--with-privsep-user” option specifies which user the unprivileged child process executes as in the privilege separation model mentioned in section two. The last option, “--with-md5-passwords”, is necessary to use shadow passwords which use an md5 hash algorithm by default.

Optional:
Include the following options if isolating user customized software configurations is desired:

```
--sysconfdir=/etc/local/openssh
--with-pid-dir=/etc/local/openssh
```

The above will neatly put the public and private keys along with the sshd process id file “sshd.pid” in the directory /etc/local/openssh away from the standard system files. Some systems will have pre-existing runtime SSH configuration files in /etc/ssh, therefore, it may be a good idea to define “--sysconfdir” to a different directory to preserve the original files.

Machines using Solaris 8 and earlier will need to specify:

```
--with-prngd-socket
```

The above indicates to read entropy from PRNGD/EGD socket file. See the “INSTALL” file included with OpenSSH for details [15].

Once configuration is complete, the typical remaining steps for building most software packages remain. As a normal user type “make” at the command prompt to compile the software. Lastly, switch user to root and run “make install” to install the software and generate a public and private rsa, dsa, and rsa1 key pair. Key generation can also be accomplished by running “make host-key”, or manually with the following commands [15]:

```
ssh-keygen -t rsa1 -f /etc/local/openssh/ssh_host_key -N ""
ssh-keygen -t rsa -f /etc/local/openssh/ssh_host_rsa_key -N ""
ssh-keygen -t dsa -f /etc/local/openssh/ssh_host_dsa_key -N ""
```

The “sshd_config” file must be set up before starting the “sshd” daemon. To enable S/key, the following should be set to “yes”:

```
# Change to no to disable s/key passwords
ChallengeResponseAuthentication yes
```

Logging is essential for forensics and intrusion detection purposes, and therefore the following may be beneficial to set:

```
SyslogFacility LOCAL7
LogLevel INFO
```

The syslog facility “LOCAL7” is optionally specified as the same facility chosen for TCP Wrappers earlier. Choosing the same syslog facility as specified for the wrappers will centralize logging, create the ability to “tail” the same log file and run programs such as “swatch” [22] for monitoring and intrusion detection purposes.

The “sshd_config” man page covers additional optional features and should be referenced for additional preferences [23].

The final step in installation is to write a startup script for the new sshd daemon. On Solaris 9, alternatively the default startup script /etc/rc2.d/S89sshd on Solaris 9 can be modified to the new configurations set by the user. The kill script /etc/rc0.d/K03sshd should also be modified to match the changes to /etc/rc2.d/S89sshd as well.

Once the sshd daemon is started, supported clients can connect to the server and authenticate using S/key after one more step:

Backtracking to TCP Wrappers, add the following to the hosts.allow file:

hosts.allow:
```
sshd: 192.168.1.0/255.255.255.0: ALLOW
```

The above configuration permits the entire 192.168.1.x network. To permit one or two hosts, the following should be adequate:

hosts.allow:
```
sshd: 192.168.1.10, 192.168.1.20: ALLOW
```

The above will allow only the IP addresses 192.168.1.10 and 192.168.1.20 permission to connect to the SSH service.

A customized OpenSSH “sshd” daemon now resides on the server with S/key and TCP Wrapper support.

7 Conclusion

Compiling software oneself creates the option to customize and add features not enabled by default, in contrast to pre-built binaries, and can improve the functionality of the software. Configuring and installing OpenSSH with TCP

Wrappers and S/key support makes it an even stronger tool by reinforcing it with additional layers of security mechanisms. Performing a custom TCP Wrappers build creates the flexibility to change default log file destinations in order to centralize logging with OpenSSH. The possibilities of customization while compiling software such as OpenSSH and TCP Wrappers is not limited to what is stated in this paper. The administrator opens numerous possibilities for elegant solutions by understanding what each feature is in a software package, and building the software to his or her preference.

References

[1] "SSH1 and SSH2 protocol hostkey change 'man-in-the-middle' attack" 22 July 2002
URL: http://www.iss.net/security_center/static/9652.php (05 Jan. 2003)

[2] "OpenSSH homepage" version 1.166 04. Mar. 2003
URL: http://www.openssh.org (10 Mar. 2003)

[3] Provos, Niels. "Privilege Separated OpenSSH" 14 Jan. 2003
URL: http://www.citi.umich.edu/u/provos/ssh/privsep.html (27 Dec. 2002)

[4] Lecompte, Christian. "Overview of S/key usage with OpenBSD" 12 July 2001)
URL: http://www.sans.org/rr/authentic/skey.php (8 Mar. 2003)

[5] Haller, N., Metz, C. "A One-time Password System". RFC 1938. May 1996
URL: http://www.ietf.org/rfc/rfc1938.txt (8 Mar. 2003)

[6] Venema, Wieste. "'README' TCP Wrapper package IPv6 version by Casper Dik" 21 Mar. 1997

[7] Venema, Wieste. "www.porcupine.org"
URL: http://www.porcupine.org/ (8 Mar. 2003)

[8] Venema, Wieste. "Wieste's collection of tools and papers"
URL: ftp://ftp.porcupine.org/pub/security/index.html (8 Mar. 2003)

[9] "S/Key Port"
URL: http://www.sparc.spb.su/solaris/skey (27 Dec. 2002)

[10] Roelofs, Greg. and Gailly, Jean-loup. "zlib Home Site"
URL: http://www.gzip.org/zlib (27 Dec. 2002)

[11] Engelschall, Ralf. S. "OpenSSL: The Opensource Toolkit for SSL/TLS" (19 Feb. 2003)
URL: http://www.openssl.org (27 Dec. 2002)

[12] “Perl Mongers”
URL: http://www.perl.org (27 Dec. 2002)

[13] Jaenicke, Lutz. “PRNGD – Pseudo Random Number Generation Daemon”
URL: http://www.aet.tu-cottbus.de/personen/jaenicke/postfix_tls/prngd.html (27 Dec. 2002)

[14] Warner, Bruce. “EGD: The Entropy Gathering Daemon” 25 July 2002
URL: http://egd.sourceforge.net (27 Dec. 2002)

[15] “’INSTALL’ file openssh version 3.5p1” 25 Jul. 2002
URL: ftp://ftp.ca.openbsd.org/pub/OpenBSD/OpenSSH/portable/INSTALL (27 Dec. 2002)

[16] Karn, Phil., Haller, Neil M.., Walden, John S.., and Chasin, Scott. “BSD man page – SKEYINIT(1) “ 24 Feb. 1998

[17] “Installing TCP Wrappers on Solaris 7 and Solaris 8” 19 July 2000
URL: http://www.kempston.net/solaris/tcpwrappers.html (8 Mar. 2003)

[18] Venema, Wieste. “’Makefile’ TCP Wrapper package IPv6 version by Casper Dik” 21 Mar. 1997

[19] “Syslog.conf (5) – Syslogd(8) configuration file”
URL: http://www.gsp.com/cgi-bin/man.cgi?section=5&topic=syslog.conf (8 Mar. 2003)

[20] “man page – inetd.conf”
URL: “http://campuscgi.princeton.edu/man?inetd.conf” (8 Mar. 2003)

[21] Venema, Wieste. “Man page – hosts_options(5)” 28 Dec. 1994

[22] “SWATCH: The Simple WATCHer of Logfiles”
URL: http://swatch.sourcefore.net (8 Mar. 2003)

[23] “Manpage of sshd_config”
URL: http://www.fifi.org/cgi-bin/man2html/usr/share/man/man5/sshd_config.5.gz (8 Mar. 2003)

Last Updated: July 26th, 2017

Upcoming Training

SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
Community SANS Omaha SEC401*	Omaha, NE	Aug 14, 2017 - Aug 19, 2017	Community SANS
Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style	Virginia Beach, VA	Aug 21, 2017 - Aug 26, 2017	vLive
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
Community SANS Pasadena SEC401 @ NASA	Pasadena, CA	Aug 23, 2017 - Aug 30, 2017	Community SANS
Mentor Session - SEC401	Minneapolis, MN	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
Mentor Session - SEC401	Edmonton, AB	Sep 06, 2017 - Oct 18, 2017	Mentor
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Mentor Session - SEC401	Ventura, CA	Sep 11, 2017 - Oct 12, 2017	Mentor
Community SANS Albany SEC401	Albany, NY	Sep 11, 2017 - Sep 16, 2017	Community SANS
Community SANS Columbia SEC401	Columbia, MD	Sep 18, 2017 - Sep 23, 2017	Community SANS
Community SANS Dallas SEC401	Dallas, TX	Sep 18, 2017 - Sep 23, 2017	Community SANS
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, Denmark	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Boise SEC401	Boise, ID	Sep 25, 2017 - Sep 30, 2017	Community SANS
Baltimore Fall 2017 - SEC401: Security Essentials Bootcamp Style	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
Community SANS New York SEC401	New York, NY	Sep 25, 2017 - Sep 30, 2017	Community SANS
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Sacramento SEC401	Sacramento, CA	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
Community SANS Charleston SEC401	Charleston, SC	Oct 02, 2017 - Oct 07, 2017	Community SANS
Mentor Session - SEC401	Arlington, VA	Oct 04, 2017 - Nov 15, 2017	Mentor
SANS Phoenix-Mesa 2017	Mesa, AZ	Oct 09, 2017 - Oct 14, 2017	Live Event