Overview of S/Key usage with OpenBSD

Christian Lecompte

Overview of S/Key usage with OpenBSD
GSEC Practical Assignment Version 1.2e
Christian Lecompte
July 12, 2001

Introduction

I have been using Open Source Software for quite a while now. Recently I mostly use The OpenBSD Operating System. In my day-to-day work I do a lot of remote administration, well who doesn't? I was getting curious about S/Key usage and integration with the OS for a while, and decided it was a very good time to give it a good test drive.

During my tests I used almost exclusively Open Source software. I did use some closed source programs under MS Windows but those were freely available with the OS (telnet.exe, ftp.exe). All other software used and discussed in this paper are Open Source under various licenses. You will witness that you don't need to pay heavy licenses fees to have decent remote access solution.

-Why OpenBSD?

Why not? I do know there are many other free operating systems to choose from, especially the more well known GNU/Linux OS. For a security conscious systems administrator I think that OpenBSD [1] stands out for some reasons. First, it's the only Open Source OS that aims at securing the system with good programming methods [2], which means that security is the number one priority. Secondly they started and now are maintaining the OpenSSH project. There is another good reason to use it it's free!

- What is S/Key?

S/Key is a One Time password authentication system used for computer logins [3]. S/Key is a good tool against replay attacks on insecure networks meaning that if someone "sniffs" your password when you log in by telnet they won't be able to re-use the password. S/Key was primarily developed by Bellcore, which owns the trademark, and is installed by default on OpenBSD 2.9 (the latest release).

- Enabling S/Key on OpenBSD

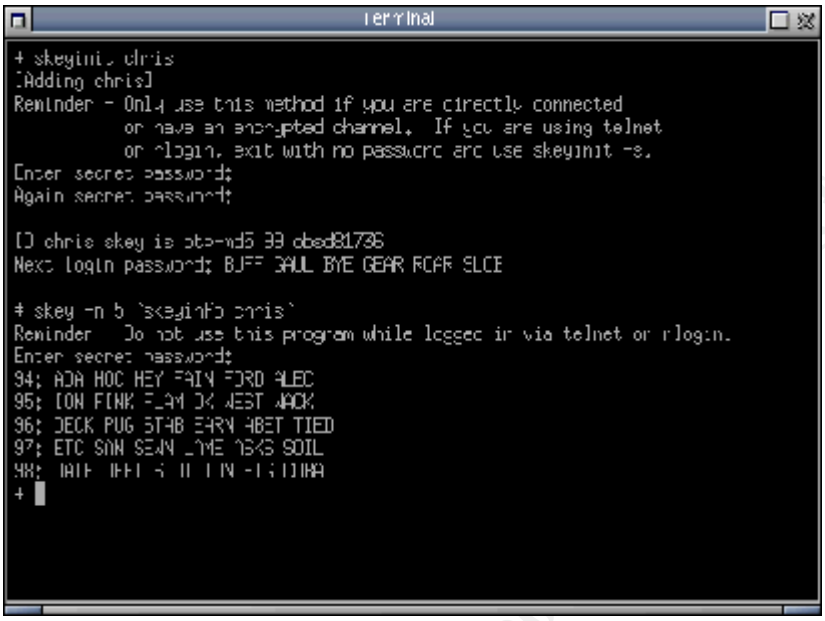
The S/Key section of the OpenBSD FAQ is excellent at explaining how to start using S/Key [4]. First of all it is recommended to make sure that the file /etc/skeykeys exists. This is the main file for S/Key utilization. This is done by typing this command as the root user:

```
# touch /etc/skeykeys
```

Now enabling S/key for a user is incredibly easy. At the command prompt type:

skeyinit user-name

You will be asked to type in your system password if you're logged in as the target user but not if you do it as root. You will be prompted next for an S/Key pass-phrase. This new pass-phrase will be the key to access your one-time passwords. For security reasons the password has to be a minimum of 10 characters. This procedure should never be done over an insecure channel. Please do it over SSH or at the system's console. Please refer to skey's man page for skeyinit usage over an insecure channel.



Let's take a closer look at the session image above. The superuser enabled S/Key for user chris. He then entered a secret pass-phrase. At that point the system printed:

ID chris skey is otp-md5 99 obsd81736
Next login password: BUFF GAUL BYE GEAR ROAR SLOB

The first line gives out a lot of information. First it tells us that chris' key was created with the md5 hash algorithm (it's the default). The number 99 means that the user has a sequence of 99 passwords. Please note that the hash algorithm and the sequence number can be changed while creating the S/Key for the user. The last information is the user's key. It will be needed to print out the next login password.

Now lets look at the following command and resulting output:

skey -n 5 `skeyinfo chris`

This is the command you want to type in when you need your next one-time password. The argument "-n" is optional and tells S/Key how many passwords we want to see, when omitted it defaults to 1. This way you could even print your whole password list, but be very careful on how you keep this information, please remember that it could be taken from you. Let me suggest

that if you have to keep this password list, encrypt the file or if you have to print it, lock it up! If you enter a bad pass-phrase you will be given a bad list of one-time passwords, and without any warning. This is a feature not a bug.

The command "skeyinfo chris" outputs the sequence number followed by the key so if we run it all by itself, this is what we would see:

```
# skeyinfo chris
98 obsd81736
```

During my testing a bug occurred on more than one of my test servers. Take another look at the picture above. If you can find the error I congratulate you. If not, here it is. When we ran skeyinit the "Next login password" differs from password number 98 in the list underneath. These two passwords should be the same. As I said earlier, I replicated this bug on more than one machine. I then submitted the bug to the OpenBSD team and am waiting for it to be fixed. In the meanwhile, execute this command " skey `skeyinfo user ` " after initialization to have the right password. It seems that skeyinit outputs a wrong sequence number.

We now used skeyinit with the default options (md5 hash algorithm and default sequence number: 100), which are quite sufficient for our demonstration puposes. To change these settings please consult skeyinit's man page.

That's it! S/Key is activated for your user. It is now possible to access the system with it. You can now login at the console, SSH or even ftp and Telnet if you enable them. Isn't S/Key easy to setup? We can now start managing S/Key.

- Managing S/Key

There are some tools that help manage S/Key. Now we will discuss the skeyaudit and skeyprune programs. We will also take a look at the /etc/skey.access configuration file.

/usr/bin/skeyaudit:

This is the tool to use to warn users if their s/key will soon expire. The program has some interesting options. If ran with the "-a" argument as root it will warn all users, by email, which have less keys in their sequence number left than the given limit. This is a nice option to run periodically as a cron job. If given the "-I" argument the warning will be given to the standard output instead of mailing the user, which can be a nice way to warn a logged in user (i.e you can add this entry to login scripts to warn users who have a sequence number smaller than 10: /usr/bin/skeyaudit -i -l 10).

The skeyaudit warning message suggests running "skeyinit -s" to reinitialize S/Key for the user. This procedure is tricky because it's also the "secure mode" for insecure channels. To reset S/Key for a user with running skeyinit without the "-s" can work just as well. It will update the user's information in /etc/skeykeys with a new sequence number, and if you wish it, you could even change your user's S/Key pass-phrase. Only use "-s" when connected through an insecure

channel, and read skeyinit's man page for further information.

/usr/bin/skeyprune:

This Perl script /usr/bin/skeyprune cleans out the /etc/skeykeys file of old entries. The skeyprune man page mentions that when skeyprune is invoked the entries zeroed by skeyinit are removed, but this is no the case. After some tests I noticed that it only removes entries older than value given at the command line (i.e the command "/usr/bin/skeyprune 100" will remove entries that are older than 100 days).

/etc/skey.access

After reading the OpenBSD FAQ I was lead to believe you could limit users login access through S/Key from the skey.access configuration file. After some unsuccessful tryouts at manipulating the file (once created) I wasn't able to change S/Key's behavior. This is when I asked the misc@openbsd.org mailing list about my problem. Mr. Todd Miller (an OpenBSD developer) told me that OpenBSD didn't support the configuration file [5] [6].

Upon response, I submitted the information to the FAQ maintainer, requesting for it to be withdrawn. As a result, it has now been removed from the web site.

- Console login

To try out our new authorization scheme we will first test it at the console. We need to log in with our user-name followed by the password "s/key" (without the double-quotes). We will be prompted next for our one-time password. We type in our assigned password to login the system. The first thing to note is that the traditional login/password combo is not disabled once the user has S/Key running, you may choose between these two possibilities. This is also true for some remote connections (i.e. telnet and SSH).

- Telnet & FTP

Telnet acts the same way as the console login so you need to enter "s/key" as the password followed by your one-time password. Ftp differs as it can accept the system or the one-time password directly, without user interaction. This behavior holds true with all tested clients even for MS Windows' ftp and telnet clients.

Fun with FTP and S/Key

With ftpd's nice way of handling one-time passwords, you could implement very easily a one-time access to your FTP server. This could be a very convenient way to share files without disrupting the security of your system. While enabling and configuring ftpd is beyond the scope of this document, please add the user-name with ftp access to the /etc/ftpchroot configuration file if you are considering doing it [7]. This will limit your user to a chroot environment (the user's home directory).

- SSH

Next we shall look into S/Key utilization with SSH, we'll cover both SSH1 and SSH2 server and client configurations. Also we will look at PuTTY a free SSH/Telnet client for MS Windows.

SSH Daemon

The SSH Daemon (sshd) configurations are done through the /etc/sshd_config file. In the default configuration SSH1 and SSH2 are enabled. It checks first for Public key authentication, and if it doesn't exist, it will prompt the user for the password. If the password authentication is unsuccessful (3 times) it will ask for the S/Key one-time password if it's initialized for the user. This default setup may not be ideal for S/Key only login scheme. It is possible to only enable S/Key for SSH logins with these configurations:

/etc/sshd_config:

```
RSAAuthentication no              # disables public key logins for ssh1
PubkeyAuthentication no           # disables public key logins for ssh2
PasswordAuthentication no         # disables password logins
KbdInteractiveAuthentication yes  # enables s/key logins
```

This is not the complete configuration file. These are only the fields needed to be set to make sure only S/Key login is enabled. You should not need to modify other fields. After modifying the configuration file you will need to send sshd a SIGHUP signal to have the new configurations enabled. For more information on sshd configuration you can consult the sshd man page [8].

SSH client

It is possible to go straight to S/Key login with some configurations on the client side. Even if the server has many other options available for logins (public-key, password) you can still go straight to your one-time password with these configurations:

OpenSSH client:

You can add this to your ~user/.ssh/config file :

```
PasswordAuthentication no              # disable password authentication
SkeyAuthentication yes                 # enable skey
KBDInteractiveAuthentication yes       # needed for ssh2 only with older OpenSSH 2.3
clients
```

These settings were tested on an OpenBSD 2.9 x86 station (native OpenSSH 2.9), a Red Hat Linux 7.1 x86 station (OpenSSH 2.5 port) and on a Yellow Dog Linux 1.2 PPC station

(OpenSSH 2.3 port). The KBDInteractiveAuthentication configuration line was only needed for the Linux PPC station running an older version of the OpenSSH port (2.3) to connect via SSH2 to the OpenBSD server.

If you don't want to make any changes to your configuration file, you can also give arguments on the command line instead. Here are some examples:

OpenSSH 2.3 clients

ssh1: ssh user@host -o 'SkeyAuthentication yes'
ssh2: ssh -2 user@host -o 'PasswordAuthentication no' -o 'KBDInteractiveAuthentication yes'

OpenSSH 2.5 & 2.9 clients

ssh1: ssh -1 user@host -o 'ChallengeresponseAuthentication yes'
ssh2: ssh -2 user@host -o 'PreferredAuthentications keyboard-interactive'

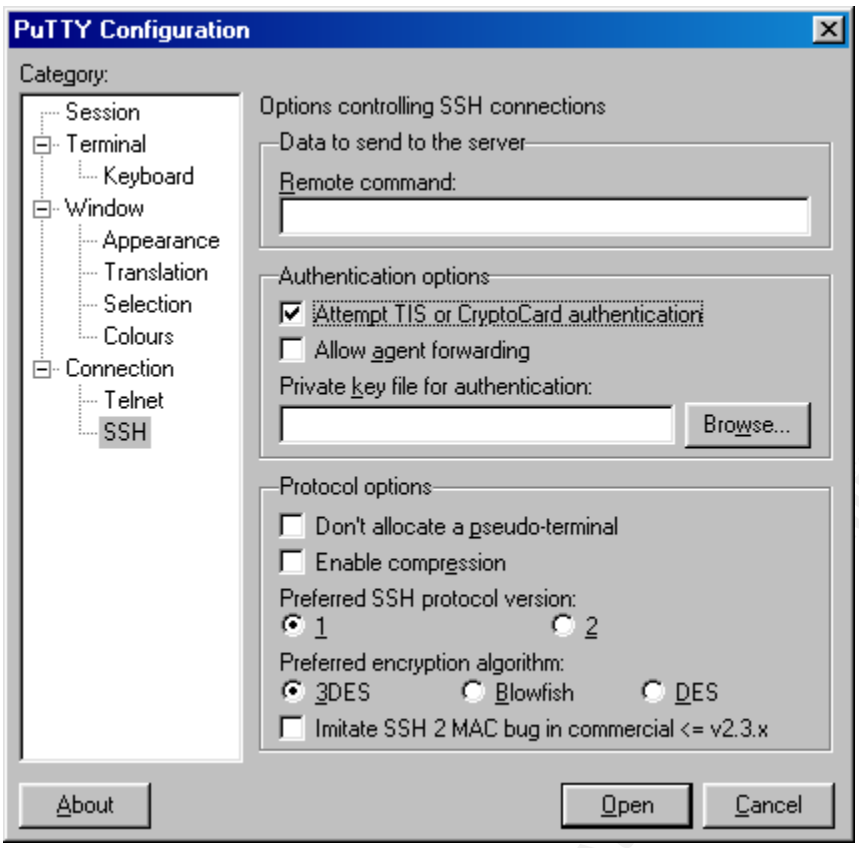
I'm sure there are some other ways to be prompted first with S/Key when connecting to an SSH server, but these methods work. You should read the ssh man page to get more details about your ssh client. Some of these methods take for granted that your client's SSH public key is not stored on the SSH server.

PuTTY SSH client

PuTTY [9] is a free alternative for telnet and SSH clients, it runs on Microsoft Windows. PuTTY can be used to log in our OpenBSD server using telnet, SSH1 and SSH2. S/Key functionality however is limited to telnet and SSH1.

Using S/Key in a telnet session in PuTTY is the same way as other regular telnet clients, you need to give the "s/key" password to be prompted for your one-time password.

To use S/Key with SSH1 all you need to do is check the box "Attempt TIS or CryptoCard authentication" as well as making sure you will be using SSH1. Please take a look at the image under for reference.



After noticing that S/Key wasn't working with SSH2, I asked the author if it was meant to be supported. S/Key support, while not active for SSH2, is being considered by the PuTTY author. He told me in an email discussion that it shouldn't be too much of a problem to add the functionality. Even with this informal discussion, there is no official stand from the developers about S/Key support for SSH2. So if you need it you may have to look some elsewhere.

- Disabling S/Key

Disabling S/Key can be done in two ways: user level or system wide. Both methods will in a way or another after the /etc/skeykeys file. To disable S/Key for a user you can edit the file to remove the line starting with the user-name. You can also put a # at the beginning of the line to comment out the user. This way the user could be re-activated without going through the skeyinit process. It could continue where it was left off. Removal of the /etc/skeykeys file will disable S/Key for all users.

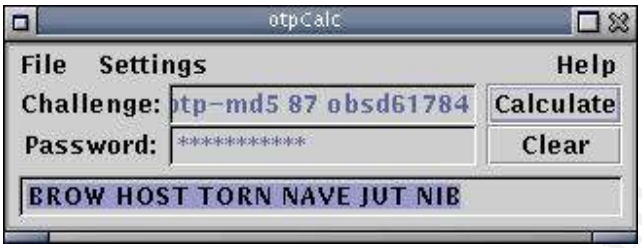
- BSD authentication (future implementation)

The OpenBSD team added the BSD authentication system to the OS a short time after the release of OpenBSD 2.9. This new system provides new methods of handling authentication. This new feature gives us new ways to deal with S/Key and other authorization schemes. It could be possible to be automatically prompted for our S/Key one-time password in a Telnet session. This information was given to me in response to a question I asked on the misc@openbsd.org mailing

list [10]. This gives us a more integrated way in dealing with S/Key. This new feature is not included in OpenBSD 2.9. Please note that at the time of this writing, this option is only available in the "-current" version of the OS. This is the developers' version, which can be quite unstable at times and should never be installed on a production system. You should only install this to test new functionality and to help report bugs. Unfortunately, during my testing I didn't have any hardware to run OpenBSD -current. So I couldn't test this new functionality. It should be available in OpenBSD 3.0.

- OtpCalc (one-time password calculator)

OtpCalc[11] is a nice little piece of free software. It permits you to calculate your next one-time password with your challenge (from the server) and your S/Key pass-phrase. This program available by source only needs an ANSI C compiler (gcc) and GTK 1.2 to compile. Here's a glimpse of the program in action:



OtpCalc can enhance you overall S/Key usage security. You wouldn't need to keep a list of your one-time passwords. You can calculate it when needed.

Conclusion

Testing S/Key functionality with OpenBSD was enjoyable and very educating. At times the implementation feels rock solid, but there were some problems especially with skeyinit printing out a bad first one-time password, or the bad documentation about the "supported" /etc/skey.access configuration file. Setting these problems aside, S/Key was very functional for day-to-day work for every type of connection I used for my servers. I hope that the information given in this paper will help others save time while trying to use it with OpenBSD. I'm also very anxious to experiment with this new BSD authentication scheme, especially if it gives out better integration for telnet and console logins.

Also, please take the time to read the OpenBSD's man pages they are excellent. Don't hesitate to use them in time of need. I have found many answers to my questions in there.

Let me conclude by adding that I do know some may think that using S/Key with SSH could be overkill. I disagree! It adds another level of security to this already good remote connection method. It can help avoid weak passwords for those who think that using an encrypted channel is a bulletproof solution by itself. Limiting SSH logins only to S/Key authentication can be a very solid way to do some serious remote administration.

References

[1] "OpenBSD home page". version 1.343. 08 July 2001.
URL: http://www.openbsd.org/ (9 July 2001)

[2] "OpenBSD security page". version 1.179. 02 July 2001.
URL: http://www.openbsd.org/security.html (9 July 2001)

[3] Haller, N and Metz, C. "A One-time Password System". RFC 1938. May 1996
URL: http://www.ietf.org/rfc/rfc1938.txt(2 July 2001)

[4] "S/Key section". The OpenBSD FAQ. version 1.66. 6 July 2001
URL: http://www.openbsd.org/faq/faq8.html#8.10 (5 July 2001)

[5] Lecompte, Christian "/etc/skey.access on 2.9". 5 July 2001
URL: http://www.monkey.org/cgi-bin/wilma_hiliter/openbsd-misc/0107/msg00380.html

[6] Miller, Todd C. "Re: /etc/skey.access on 2.9". 5 July 2001
URL: http://www.monkey.org/cgi-bin/wilma_hiliter/openbsd-misc/0107/msg00382.html

[7] "Confining users to their home dir's in ftpd(8)" OpenBSD FAQ. version 1.54. 9 June 2001
URL: http://www.openbsd.org/faq/faq10.html#10.13 (7 July 2001)

[8}"sshd - OpenSSH SSH daemon" OpenBSD manual pages. 25 September 1999
URL: http://www.openbsd.org/cgi-bin/man.cgi?query=sshd&apropos=0&sektion=0&manpath=OpenBSD+Current&arch=i386&format=html (5 July 2001)

[9] Tatham, Simon. "PuTTY A Free Win32 Telnet/SSH Client". 28 February 2001
URL: http://www.chiark.greenend.org.uk/~sgtatham/putty/ (25 June 2001)

[10] Insulander, Hans. "Re: permitting only S/Key login for telnet". 27 June 2001
URL: http://www.monkey.org/openbsd/archive/misc/0106/msg02168.html (4 July 2001)

[11] "OtpCalc - an OTP and S/Key calculator for X"
URL: http://original.killa.net/infosec/otpCalc/ (10 July 2001)