SIO Lab
SSH and OTP Authentication

version 1.0

Authors: João Paulo Barraca, Vitor Cunha, Alfredo Matos

Version log:

- 1.1: Added TOTP / HOTP authentication.
- 1.0: Initial version

1 Introduction

In this guide we will explore authentication mechanisms by configuring SSH (Secure Shell) secure sessions. SSH is an IETF standard comprised of Authentication, Connection and Transport standards, composed by several RFCs (RFC 4250, RFC 4251, RFC 4252, RFC 4253 and RFC 4254), and is a common protocol used for several purposes, such as remote shells, git repository data transport, and many other interesting scenarios.

It supports several authentication mechanisms, from which we will explore simple username and password authentication, asymmetric key authentication, and finally One Time Passwords (OTP).

2 Setup

To execute this lab you need two (virtual) machines (your virtual machine and a container will be sufficient). One system (the container), `M1`, will have the role of server. The other, `M2`, will have the role of client, for which you can use your virtual machine.

2.1 Server Configuration

2.1.1 Container

Create the container to run the server.

```
# Create the server container
vm:~$ lxc launch images:debian/bookworm server
```

2.1.2 Services

Make sure you log into the server, create a user that you can use to log remotely into the machine (we suggest user sio, password sio), and install the required services.

```
# Log into the server
vm:~$ lxc shell server
server:~$ adduser sio
```

Check if the `Telnet`and `SSH`services are installed and enabled in the server computer (M1). For that, use the following command to check if the TCP ports used by those services are available to accept connections.

```
server:~$ netstat -atn
```

With the following commands you may install and configure the SSH and Telnet services in a Linux computer. Use those that are adequate to install the services missing in M1.

Start by updating the repositories with:

```
server:~$ apt update
```

Install the Telnet service (`telnetd`), using:

```
server:~$ apt install inetutils-telnetd
```

Edit `/etc/inetd.conf`, and enable (uncomment) the telnet line (23):

```
# /etc/inetd.conf
telnet stream tcp nowait root /usr/sbin/tcpd /usr/sbin/telnetd
```

Then restart the Telnet service, with the command:

```
server:~$ systemctl restart inetutils-inetd
```

Install the SSH service with the following command:

```
server:~$ apt install openssh-server
```

If you need to restart the SSH service, you may use the command:

```
server:~$ systemctl restart ssh # or systemctl status ssh to check current status
```

Check that both services are up and running.

You also need to install `Wireshark` because you will need to capture and analyse packets exchanged between M1 and M2 computers.

2.2 Client Configuration

By default, SSH and Telnet client applications come installed in most Linux distributions. In case you need to install any of them, you may the following commands (use the ones that apply):

```
sudo apt install telnet ssh
```

Note: A client application for Windows is `PuTTY`, available at http://www.chiark.greenend.org.uk/~sgtatham/putty. It is both a Telnet and SSH client.

3 Inspecting network traffic

One of the toolkits under an engineer's belt is the use of traffic capturing utilities to find out what is going on. This can be very useful for security purposes.

In the client computer, using `Wireshark` start a capture of the traffic exchanged between the server and the client. Apply a filter to only observe `Telnet` and `SSH` packets, by applying the following rule in the filter window:

```
telnet || ssh
```

With this filter applied, `Wireshark` will show all the SSH and Telnet packets exchanged between server and client computers.

3.1 Inspecting `Telnet` traffic

In the client computer (M2), with an active traffic capture, start a `Telnet` session to the server computer (M1). First, get the ip address of the container by reading it from `lxc list` or `lxc info server` and then telnet into the server (when asked, use the credentials you created before):

```
telnet IP_ADDRESS
```

After login, list the contents of the root folder, using the command:

```
ls /
```

Close the `Telnet` session, stop the `Wireshark` capture and analyse the captured packets. Can you observe, in the captured packets, the contents of the root folder you listed above?

Still analysing the captured Telnet traffic, can you find the password you used to login when starting the Telnet session in the server computer (M1)?

What do you conclude regarding the security of the Telnet protocol?

3.2 Inspecting SSH traffic

In the client computer start a new capture and apply a filter to only show the captured SSH packets.

In the client computer (M2) start an SSH session with the server computer (M1). In case it is the first connection to that server, the server will authenticate presenting its public key and you are asked if you trust the key and want to save it. Say yes to proceed. Login into the server and list the contents of the root folder using the command:

```
ls /
```

Close the `Telnet` session, stop the `Wireshark` capture and analyse the captured packets. Can you observe any data in the packets. What do you conclude about the general security of SSH, comparing to Telnet?

4 Authentication in SSH

In this section you are going to analyse how the server and clients authenticate in the SSH protocol. It is assumed that SSH configuration files are in their original state, i.e., no change has been made to the SSH configuration.

4.1 Server autentication

In the first interaction of an SSH client with an SSH server, the client must validate the server identity. For that purpose, the SSH server send its public key to the client and the respective fingerprint in presented to the user, so they can verify it and indicate if the key is correct or not, i.e., if the user trusts the server to which a session is being started, or not. For this, the user must be in the possession of the public key fingerprint of the SSH server to which they want to connect (she should be given the public key fingerprint when her account was created). To get the fingerprint of the SSH server public key, issue the following command in the server computer, and write the fingerprint value:

```
    sudo ssh-keygen -l -f /etc/ssh/ssh_host_ed25519_key.pub
```

In the client computer (M2) eliminate the public keys that you possibly stored from previous SSH connections. For that purpose, use the following command:

```
rm ~/.ssh/known_hosts
```

From the client computer, start an SSH session with the server, using the following command, where *username* (e.g. sio, if you created) is your account name in the server and *host* is the name or IP address of the computer with the SSH server (M1):

```
ssh username@host # ssh sio@LXC_SERVER_IP
```

The user in the client computer will be presented with a message similar to the one shown bellow, asking if the user trusts the identity of the SSH server they are trying to connect to. Compare the presented public key fingerprint with the fingerprint you get in the SSH server. If they are equal, then you are trying to connect to the SSH server you intent to connect and you must answer `yes`.

```
The authenticity of host '192.168.56.1 (192.168.56.1)' can't be established.
RSA key fingerprint is 6a:de:e0:af:56:f8:0c:04:11:5b:ef:4d:49:ad:09:23.
Are you sure you want to continue connecting (yes/no)? yes
```

After the confirmation the client stores the public key in the `~/.ssh/known_hosts` file and the user is never again requested to confirm that SSH server public key, as long as the server keeps presenting the same public key.

Proceed to start the SSH session, by introducing your password to authenticate yourself to the SSH server. Login and password is the default mechanism for user authentication with SSH servers.

4.2 User authentication with an assymetric key pair

In the client computer you must generate an asymmetric key pair to use for client authentication when starting a session in an SSH server. For that, use the following command:

```
ssh-keygen -t rsa
```

Press `Enter` to accept the default name for the file where the key pair will be stored, and define a password for key protection.

Now, you need to install your public key in the SSH server, in order the server can use it to authenticate you when starting SSH sessions. For that purpose, use the following command, where `~/.ssh/id_rsa.pub` is the default filename used to store client keys (replace it with the name of the file where you stored your key pair, in case you haven't used the default file name), and `username` and `server` are the name of the account and the name of the computer where you want to install your public key:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub <username>@<server>
```

This command stores your public key in the `~/.ssh/authorizedkeys` file in the home folder of the user account in the server computer. This public key will be used to verify the user possession of the corresponding private key, whenever the user starts an SSH session with the server.

In the client computer, start a new SSH session to the SSH server in M1. Notice that your public key is used for your authentication and you are not requested to introduce your password.

Compare public key authentication with password authentication when starting an SSH session and analyse the advantages and disadvantages from one in relation to the other.

Public key authentication on SSH is substantially different from the authentication in SSL protocol. Explain the differences.

Do you think it is adequate to use the SSH public key authentication to authenticate Web servers in the Internet, for example to authenticate Google or Facebook servers? Explain.

4.3 User authentication with one-time passwords

Situations may exist where asymmetric key pairs might not be the most adequate mechanism to authenticate users when establishing SSH sessions. For example, one such situation may occur when the user uses a shared computer and fears that its key pair can be compromised. In such situations, one-time passwords may be the most adequate mechanism for user authentication when establishing an SSH session with a remote server, since one-time passwords can only be used one single time (they are not reusable), and therefore no security mechanisms are needed to secure its use.

4.3.1 Server configuration

To use one-time passwords in SSH, you need to install `OTPW` in the server. Use the following command:

```
sudo apt-get install libpam-otpw otpw-bin
```

Then you need to configure PAM (*Plugable authentication Module*) module of SSH. For that, you must edit the `/etc/pam.d/sshd` configuration file (using vim or gedit, for example). You must disable password authentication, by commenting the following line (by placing a # character in the beginning of the line):

```
#@include common-auth
```

Then, you must add the following two lines to enable user authentication using one-time passwords:

```
auth    required    pam_otpw.so
session optional    pam_otpw.so
```

Then you must configure the SSH server to accept one-time passwords. For that, you must edit the `/etc/ssh/sshd_config` file to enable the following three parameters (check that none of these parameters is duplicated, to avoid server failures when starting):

```
UsePrivilegeSeparation          yes
ChallengeResponseAuthentication yes
UsePAM                          yes
```

You also need to disable password authentication. However, we will allow public key authentication to prevent the possibility of all one-time passwords were used. For that, edit again the `/etc/ssh/sshd_config` file and check the following two parameters have the indicated values:

```
PubkeyAuthentication    yes
PasswordAuthentication  no
```

SSH server configuration is completed, and must be restarted.

4.3.2 One-time passwords generation

One-time passwords for user authentication must be generated and given to the user, in order they can use to authenticate when establishing an SSH session. To generate the user one-time passwords, the user must be logged in the SSH server (not has root) and must execute the following command, where `fname.txt` is a name for the file to store the generated one-time passwords. When executing the command, the user will be requested to introduce a prefix for the one-time passwords, that the user must memorize because it will be requested, together with the one-time password, when authenticating.

```
otpw-gen > fname.txt
```

Look at the contents of the file with the generated one-time passwords. Note that the one-time passwords are numbered and that each occupies two columns to make them easy to read. When the user is requested to introduce a one-time password for authentication, they must concatenate the text in the two columns, i.e., the blank space between the two columns must not be introduced.

The user must print the file to bring the one-time passwords with her, in order they can use them whenever they establishe a new SSH session with the server.

4.3.3 Using one-time passwords

The establishment of a new SSH session is done with the same command as indicated elsewhere above in this guide, but now, the server will request one of the one-time passwords the user generated. The requested one-time password is identified by its number in a format similar to:

```
Password 123:
```

The full one-time password to be introduced is composed by two parts: the prefix (defined by the user when they generated the one-time passwords) and the identified one-time password that the user must read from the file containing the one-passwords they generated. The user must concatenate the prefix with the requested one-time password. If the correct password is introduced, SSH session is established.

4.4 2FA with Google Authenticator (TOTP/HOTP)

We can also complement the login with 2-factor authentication (2FA). This is simple to achieve with SSH and a complementary time based one time passwords, such as TOTP or HOTP. This can be done with Google Authenticator for example.

The setup is similar to what was done before, by setting up SSH and PAM. So, please ensure the following:

1. Comment/remove the information from OTPW (previous exercise)
2. Install Google Authenticator from the Play/App Store

Now, let's jump into setting up the server and the account.

4.4.1 Server Authentication

```
apt install libpam-google-authenticator
```

With the user you want to setup

```
su sio
google-authenticator # go through the setup process
```

Scan the information with Google Authenticator, and lets enable it on SSHD.

Then you must configure the SSH server to accept one-time passwords. For that, you must edit the `/etc/ssh/sshd_config` file to enable the following three parameters (check that none of these parameters is duplicated, to avoid server failures when starting):

```
UsePAM  yes
ChallengeResponseAuthentication yes
```

Then you need to configure PAM (*Plugable authentication Module*) module of SSH. For that, you must edit the `/etc/pam.d/sshd` configuration file (using vim or gedit, for example):

```
#@include common-auth
auth   required   pam_google_authenticator.so
```

Then just restart SSH, and it should be working. 2FA.

5 Acknowledgements

Originally authored by João Paulo Barraca, and Vitor Cunha

5.1 Bibliography

http://en.wikipedia.org/wiki/Secure_Shell