MCS: Identification, Authentication and Authorization 2020-21

Practical Exercises: User authentication on SSH with asymmetric credentials

April 1, 2022 Due date: no date

Changelog

- v1.0 - Initial version.

1 Introduction

The goal of these exercises is to explore the functionalities of SSH to authenticate users with asymmetric credentials. We will explore authentications with both key pairs created by the user and with key pairs where the private key is stored inside a cryptotoken with a PKCS#11 interface. For this last alternative, we will make use of a virtual smartcard and the Portuguese Citizen Card.

This work will conceptually involve two different machines: the server, where an SSH daemon runs, and a client, where the user that want to create a session on the server uses the SSH client for that end. We will assume the client is a Linux machine, but you can possibly do it on any other operating system. If using a Linux client, you can also use only one machine for the experiments we are going to perform.

2 SSH daemon

Install the `ssh` package on the server host and start the SSH daemon with the following command (for Ubuntu, it may be different for other distributions):

```
service ssh start
```

You can immediately test if it is working by running the client on the same machine:

```
ssh localhost
```

This command also shows you a (hashed) fingerprint of the daemon's public component of its authentication key pair. This information is crucial to trust on SSH daemons, since they do not use certificates (in a way, it is a bit just like in PGP, each one defines the information they trust).

3 SSH client

Install the `ssh` package on the client host.

Run the following command to initiate a session for user `username` at host `hostname`:

```
ssh username@hostname
```

Use a `username` for which you know the password at `hostname`.

Terminate this SSH session before proceeding to the next one.

In this first SSH session creation you had to accept the SSH daemon's public key provided by host `hostname` as correct. In a real exploitation scenario you should be very careful when doing it! The known (and trusted) public keys of SSH daemons are stored in text files, such as these:

```
/etc/ssh/ssh_known_hosts
\textasciitilde/.ssh/known_host
```

You can know confirm that this file was created (or updated), but you cannot know which hosts are listed in these files, since the `hostname` provided by users was hashed and constitutes the search key for the correct daemon's public key.

4 Asymmetric user authentication

In many cases it is convenient to use asymmetric credentials for user authentication. In particular, when someone administers dozens of hosts, it becomes very handy because one do not need to type the password over and over again. Also, if by mistake we are fooled by a rogue server, we do not provide it our password!

4.1 Creation of asymmetric credentials

Asymmetric credentials are created with the command `ssh-keygen`. Keys are stored in files, either using a stem name or a default name and path. The default path is `\textasciitilde/.ssh`, the default stem is the `id_type`, where `type` is currently `rsa` (the default), `dsa`, `ecdsa`, `ecdsa-sk`, `ed25519` and `ed25519-sk`. The private key is stored separately from the public one, the file for the public key has an extra `.pub` appended to the stem.

Run the following command to create a (256) bit key pair using Bernstein's Elliptic Curve 55519:

```
ssh-keygen -t eb55519
```

4.2 Setup of the public credential on the target host

In order to support the authentication of a user with an asymmetric credential, the SSH daemon will check a signature made with a private component with a public counterpart that it (daemon) knows. This public key must exist under the users' directory, in a file named `.ssh/authorized_keys`. This name is mandatory, since this file is looked upon by the SSH daemon. But this file can contain several different public keys, possibly from different types.

You can manually set up this file or you can update it using the command `ssh-copy-id`. This command performs a login into the account where you want to add a public key, and updates the relevant file. Thus, execute:

```
ssh-copy-id username@hostname
```

log in with the username/password pair and the public key setup is performed.

By default, this command will upload all the identity files that you have. Identity files are the default files containing your SSH public keys. In order to upload a unique identity, use option `-i` and the name of the file with the desired identity (public key).

Once this is done, you can use the asymmetric credentials in the next SSH session. Run the command

```
ssh username@hostname
```

provide the password to unlock the private key file and that's it!

4.3 User agent

The management of users' SSH asymmetric credentials gets simpler when being handled by a user agent, `ssh-agent`. This is an application that runs in background, by default. For starting it use the command (do not execute them!)

```
ssh-agent
```

and for terminating (killing) it, use the following command

```
ssh-agent -k
```

This application echoes a set of shell/C-shell commands that are useful for setting up (and cleaning) environment variables that help SSH commands to interact with the current user agent. Therefore, they should run as follows:

```
eval ‘ssh-agent‘
```

```
eval ‘ssh-agent -k‘
```

In order to save you the effort to always provide the password to unlock a private key, you can use the `ssh-agent` for memorizing the password. This is done with another command, `ssh-add`. This command allows you to add a new identity (private key) to the list of credentials known by the user agent. Therefore, execute

```
ssh-add
```

and this command will add all your default identities to the actual agent. For that, you need to provide the password hat unlocks them. Do not forget to initiate the user agent first.

The user agent does not maintain persistent state. Therefore, if it terminates, you need to provide again all your identities when it starts, and also the passwords to unlock them.

The communication with the agent is performed through a local communication channel, which end-point name is only accessible to the agent owner. You can see the name of it in the environment variable `SSH_AUTH_SOCK`:

```
echo $SSH_AUTH_SOCK
```

Check with

```
ls -l
```

its protections.

5 Using the Portuguese Citizen Card with SSH

The Portuguese Citizen Card (PTEID hereafter) already possesses an RSA authentication key pair that can be used with SSH. All that is necessary is to use the SSH tools combined with the PTEID PKCS#11 library.

The first thing we need is to install this library, which part of the PTEID software. Therefore, download and install in the SSH client the correct software package from `https://www.autenticacao.gov.pt/web/guest/cc-aplicacao`. After this installation, the PTEID PKCS#11 file is the file `libpteidpkcs11.so` located in directory `/usr/local/lib`.

Connect a smartcard reader to the client host, insert your PTEID card on it and run:

```
pkcs11-tool -module /usr/local/lib/libpteidpkcs11.so -O
```

This command lists the objects within the PTEID: we are interested in the key pair with the label `CITIZEN AUTHENTICATION KEY`.

The command `ssh-keygen` can be used to extract the keys from the PTEID card. However, this command extracts multiple keys, including the keys belonging to the certification chain. But this can be easily solved with some filtering, as follows:

```
ssh-keygen -D /usr/local/lib/libpteidpkcs11.so | grep AUTHENTICATION >
/.ssh/id_pteid.pub
```

Now we upload this key to the server side as before, with `ssh-copy-id`, but with an extra flag (`-f`):

```
ssh-copy-id -f -i  /.ssh/id_pteid username@hostname
```

The `-i` flag indicates the identity file to upload, while the `-f` is used to force the upload, which would not occur in this case because there is no file with the private key.

You can log in remotely with SSH now and check the presence of the PTEID public key in the file with the authorized keys. Do not forget to log out before continuing.

Finally, we can use SSH with the PTEID credentials with the following command:

```
ssh -I /usr/local/lib/libpteidpkcs11.so username@hostname
```

Note: due to several implementation details, this command will actually fail.

6 References

- Portuguese Citizen Card web site: `https://www.cartaodecidadao.pt`