Introduction à la sécurité TP 2 - Outils

Maciej Korczynski & Simon Fernandez

Novembre 2021

Objectives

- Use popular tools for system and network security

Network configuration

In this tutorial, according to your goal, you will need two network configuration :

- **NAT** : if you want the VM to be able to communicate through Internet (it is the better configuration, because it is more realistic)
- **Host-only networking** : if you want make communicate easily the host and the VM (if you have some trouble)

For the Host-only networking, you need to create a Host-only network (from the VirtualBox manager, File -> Host Network Manager...).

For the NAT configuration, you will need to use port fowarding (Devices -> Network Adapters..-> Port forwarding) to access to your services.

You can easily switch between this two configurations or setup both in parallel from the VM (Devices -> Network Adapters..)

SSH (Secure Shell)

The tool

SSH is a powerful cryptographic network protocol used to perform remote command-line through a network. SSH replaces the use of Telnet, FTP, ... by adding encryption of communication.

SSH offers a lot of use, for example :

- Remote admin
- File transfer
- Port forwarding (X11 forwarding, firewall traversal)
- VPN

OpenSSH is an open-source version of SSH. `openssh-server` is the package for the server side (listening on port 22), and `openssh-client` for the client.

Configuration file for the server is located in `/etc/ssh/sshd_config` Configuration file for the client are located in `/etc/ssh/ssh_config` and `~/.ssh/config` (if it exists)

There are some tools, related to `ssh` :

- `scp` : copy through SSH
- `sfp` : `sftp` just like `ftp` but going through the SSH
- If a graphical server is installed on the server, you can control it with the `xhost` command and the `-X` flag in the `ssh` command

Server Authentication

In this configuration, your VM will be the server and your host machine will be the client.

- Install `openssh-server` on your VM.
- Create your own 'host' keys in `/etc/ssh` :
  - Delete old keys in `/etc/ssh/` (the `ssh_host_*_key` files)
  - Create new RSA and new DSA key pairs with :

  `ssh-keygen -t rsa  -b 2048 -f /etc/ssh/safe_host_rsa_key`
  (do the same for DSA)
- Update `/etc/ssh/sshd_config` to use these new keys (directive `Hostkey`)
- Connect your client to the server : `ssh user@IP_SERVER -p PORT` (if you use NAT configuration, you will need a different port, according to your mapping)

User Authentication

There are different ways to manage user authentication :

- Use of couple login / password
- Use of public / private key (see : https://en.wikipedia.org/wiki/Public-key_cryptography) It's better to use key instead of password (passwords are insecure, crackable, easily forget on some paper in your office..).

Each client can generate a couple key public/private, with :

`ssh-keygen -t rsa -b 2048`

(it's better to use a complex pass-phrase at this point)

This will create 2 files in `~/.ssh/` : `id_rsa`, `id_rsa.pub` (or `id_dsa`) . The `.pub` file is the public key, the other one is the secret key (and by secret we mean it never ever give it to anyone). The public key can be appended to any `~/.ssh/authorized_keys` files on SSH server you want to connect to.

Practice

Create on your host a couple of key public/private, and add the public into the server. Find how to configure your ssh server to forbid connection with password (in `/etc/ssh/sshd_config`).

Bonus : I want be able to have different couples of public/private key for different SSH servers on the same machine. How create different keys in the same machine ? How I can specify which key must be used according to the server ?

Wireshark/Tcpdump

Wireshark is a network traffic analyzer. Wireshark is the graphical version of `tcpdump`. They capture each single data packet seen by the network card.

Networks are very noisy, when you look for a specific host or protocol, looking into the complete list of data can be very hard, for the convenience use filters.

You can find some example of useful filters : http://wiki.wireshark.org/CaptureFilters

For example, filter `tcp.port==80` will only captures TCP traffic on port 80 (be careful filter syntax can be little different between `tcpdump` and `wireshark`).

If you work on a system without graphical service, you can use `tcpdump` to capture packet, save it in a file (option `-w`). Then, you can send the file to a remote machine (use `scp`) and read it with Wireshark, for example :

```
(target): tcpdump tcp port 80 -w file_output
(remote): scp USER@IP_TARGET:/path/to/file/file_output
(remote): wireshark file_output
```

Practice

Find how to, in one filter (`wireshark` or `tcpdump` format), capture only SSH traffic (port 22) or traffic to `173.194.78.94 (www.google.com)`. Observe and learn how to use Wireshark, try to follow some http, ftp or ssh traffic, it will be very useful later.

Nmap

Nmap is a port scanning tool : it tries to find what ports are opened on computers and identifies services. There are several scanning methods, some are intrusive, some are hard to detect. Nmap provides a lot of options (see `man nmap`), for example :

- To detect machine on network : `nmap -sP <IP>` (IP can be one IP, or a network (for example `192.168.0.0/24`), or a range (`192.168.100-200`)
- To detect open TCP port : `nmap -sS <IP>`

Practice

Do some test with `nmap` from the host machine to your VM.

Server side: run `tcpdump` to see what is received: - `sudo tcpdump -i <interface>` to capture all the traffic going through an interface.

Client side: scan the server: - Basic scan: `nmap -sT SERVER_IP`.

How does the `-sT` scan work? Compare the output of `nmap` and the sequence of packets that were captured by `tcpdump`. What are the different scans provided by `nmap`? How do they work? Are they stealthy?

- Find how to get the `openssh` and `apache2` version of your VM with `nmap` (give command and results).

fail2ban

Fail2ban reads the logs from various servers (SSH, Apache, FTP, ...) and searches some repeated authentication errors to add an `iptables` (firewall) rule to ban the IP address of the source. It is very important if some of your services are directly accessible from Internet (a lot of zombie machine scan random computer and try to connect into it).

You can configure `fail2ban` to suit your needs and enable the filters that you find interesting.

The file `/etc/fail2ban/jail.local` defines which services `fail2ban` will be watching and custom configurations. This file (that you may have to create from scratch) has the following syntax:

```
[<filter_name>]
enabled = true
<option> = <value>
```

All filters are in `/etc/fail2ban/filter.d/`. One file per service. The default configuration of default filter is in `/etc/fail2ban/jail.conf`

Practice

Default filters

What existing filters may be interesting for our server?

Custom filters

We will create a new filter, for `DFind w00tw00t` request, a famous script kiddie request.

First, we create the filter file `apache-w00tw00t.conf` in `/etc/fail2ban/filter.d` :

```
[Definition]
failregex = ^<HOST> -.*"GET \/w00tw00t\.at\.ISC\.SANS\.DFind\:\).*".*
ignoreregex =
```

Then we add the default configuration in `/etc/fail2ban/jail.local` and enable the filter :

```
[apache-w00tw00t]
enabled = true
filter = apache-w00tw00t
action = iptables[name=Apache-w00tw00t,port=80,protocol=tcp]
banaction = iptables-allports
logpath = /var/log/apache2/access*.log
maxretry = 1
banTime = 86400
```

You can check the proper functioning of this rules with :

```
fail2ban-regex /var/log/apache2/access.log /etc/fail2ban/filter.d/apache-w00tw00t.conf
```

Bonus : Why is it useful to add a time ban ?

John the Ripper

John the Ripper is an automated password cracker. He reads files containing password hashes (like `/etc/shadow`) and tries various dictionary/algorithms based attacks to ”crack” the passwords.

Practice

Install `john` and `wfrench` (french dictionary) or any dictionnary for your native language. Change your password with a simple password (a simple french word). Then run :

```
$ john -wordlist:/usr/share/dict/french /etc/shadow
$ john -show /etc/shadow
```

You can try with some different passwords (restore your previous password at the end !).

Snort

Snort is open source network intrusion prevention system (NIPS) and network intrusion detection system (NIDS). He works as a normal sniffer (like wireshark), but collected data is used to find attacks on networks.

Collected data is processed ”live” by a huge number of rules, each rule describing potential attacks : virus attacks, web attacks, deny of services, floods, suspicious file transfers, suspicious conexions, ...

Usually snort is installed on gateways or on a specific host to which all network traffic is ”mirrored” to (switches have this functionality for example)

The principal configuration file is `/etc/snort/snort.conf`. You can change this file according to your need. Rules files are in `/etc/snort/rules/`

Practice

Launch some scan with `nmap` while `snort` is running. What happens in `/var/log/snort/snort.log` ? Take some rules from `/etc/snort/rules` and explain how they work (you will find more information in http://manual.snort.org/node27.html).

This tutorial is aimed to show you some useful tool. You will find a lot of good advice on network security in http://www.ssi.gouv.fr/fr/bonnes-pratiques/ (reading some guidebook is highly recommended)