Lab 5

DNS Poisoning, Subprocess, Regular expression, and HSTS

1. DNS Poisoning with Bettercap
   Turn on Ubuntu and Kali, and check their IPs.
   (a) On Kali, run webserver: `sudo service apache2 start`

   (b) On Kali, then run Bettercap: `sudo bettercap` to run Bettercap.

   (c) Perform ARP poisoning attack to Ubuntu. Type `set arp.spoof.fullduplex true` (Type `help arp.spoof` to know what it does.), followed by `set arp.spoof.targets <Ubuntu IP>` and `arp.spoof on`.

   (d) Perform DNS poisoning. Type `set dns.spoof.domains amazon.com`, followed by `set dns.spoof.address <Kali IP>. Lastly, dns.spoof on`.

   (e) Go to Ubuntu and run or `arp -a` to check the gateway IP. Open a browser, ***delete all the history** (everything) and access `http://www.amazon.com`. You should be redirected to the website Kali is running, instead of amazon.com.

   *Alternatively, you can type the following command on Ubuntu: `sudo resolvectl flush-caches` before opening the browser and entering `http://www.amazon.com`. You can check whether amazon.com's IP has changed by entering `ping amazon.com` on terminal.

2. Subprocess in Python
   The subprocess is a powerful Python module that allows us to run system commands in any OS including Unix/Linux to pipe input and output, and client programs.

   The subprocess module has many functions. The most basic syntax is as follows:

```
import subprocess
subprocess.run('COMMAND')
```

   Let us create a simple program that makes use of the subprocess module. Type the following code in `subprc.py` and run it.

```
import subprocess
subprocess.run('ls')
```

You can pass extra arguments for your command by modifying the above code as follows.

```
import subprocess
subprocess.run('ls -l', shell = True)
```

Now change the subproc.py to run `ifconfig` for a network interface name as user input. Type the following and run it.

```
import subprocess

interface = input('Enter interface name> ')
subprocess.run('ifconfig ' + interface, shell = True)
```

Input any interface name. What do you see?

By using `shell = True`, you can run any Unix (Linux) commands with any sequence of arguments. However, if we think about *secure coding*, this method has a drawback: Provide `eth0;ls` as an interface name to the above program. What do you get?

You actually expected to run `ifconfig interface` but your Python program also executes `ls`. This shows that executing the subprocess.call with `shell=True` is dangerous (if we are a defender). Therefore, we split the command and options into a number of elements using Python list:

```
interface = input('Enter interface name> ')
subprocess.run(['ifconfig', interface])
```

Note that this is a safe way to use the run function in subprocess.

So far, the subprocess just has *run* the Unix command and *displayed* the result on the screen. Now, we want to capture the output somehow and to process it further. In this case, you modify the last line of the above code into

```
import subprocess

interface = input('Enter interface name> ')
p=subprocess.run(['ifconfig', interface])
print(p)
```

Here, the result of the `ifconfig` command is assigned to the variable p. Run the program and see what is displayed as a result. You may have expected that you would see the ifconfig result, but what you see is

```
CompletedProcess(args=['ifconfig', 'eth0'], returncode=0).
```

This is just a set of attributes of completed process. To capture the output of the process, you need to modify the line of the above code as follows:

```
p=subprocess.run(['ifconfig', interface],
capture_output=True)
```

Rerun your code with `print(p.stdout)` at the end. You will see the result but there will be no new lines in it. This is because the captured process `p` is represented in byte. To change this to string, you need to modify `print(p.stdout)` to `print(p.stdout.decode())`.

The useful feature of subprocess is that we can *pipe* one process to another. In the following code, the output of `ifconfig` is piped (redirected) to another process to run "`grep`" to display any lines containing the words "inet".

```
import subprocess

interface = input('Enter interface name> ')
p1 = subprocess.run(['ifconfig', interface],
capture_output=True)
p2 = subprocess.run(['grep','inet'], capture_output=True,
input=p1.stdout)
print(p2.stdout.decode())
```

3. Regular expression in Python
   In the example in Task 2, we used "grep". But we want to use something more sophisticated Python module to extract strings from the output. The solution is to apply `re` (regular expression).
   First, go to https://www.w3schools.com/python/python_regex.asp and have a look at the sections on metacharacters, special sequences and sets. Then, go to https://pythex.org and derive regular expressions to get a MAC address from the `ifconfig` result: To do this, cut and paste the ifconfig result to the "Your test string" window. Your task is to find a regular expression for finding a MAC address. (Hint: Use special sequence. You don't have to find a very complex regular expression.) Finally, modify your code from Task 2 as follows (You may want to create a new code by cutting and pasting the previous code.):

```
import subprocess, re

interface = input('Enter interface name> ')
p1=subprocess.run(['ifconfig', interface],
capture_output=True)

regx = re.compile('') # enter your regular expression here
mac_addr = regx.search(p1.stdout.decode())

print(mac_addr.group())
```

*Note 1: `group()` is a method from `re` class, which returns one or more subgroups of the match.
**Note 2: Alternatively, you can change sthe fourth line to `p1=subprocess.run(['ifconfig', interface], capture_output=True, text = True)` not to use decode() method. Then you can directly use `p1.stdout` as a string. (The same applies to the code in Task 1)

4. Applying subprocess further and making your python executable.
   We sometimes need to make our Python program executable. To do this, we add shebang line at the beginning of the code:

   ```
   #!/usr/bin/env python
   ```

   Then, issue `chmod +x yourfile.py` or `chmod 755 yourfile.py`. You can execute it by issuing `./yourfile.py` on terminal.

   Your task is to write a Python program using subprocess to make the `nmap` to take the target IP from the user as input. Then output any `syn` scan results concerning the port 3306 only. (That is, display any strings that contain “3306”.) Use the grep command not regular expression. Assume that the target server is Metasploitable. Make your program executable.

5. [Investigation] HSTS in practice

   The HSTS list used by Chrome can be checked at https://hstspreload.org/
   Try to enter facebook.com, twitter.com, www.amazon.com (interestingly, “amazon.com” is not in the list.) You can download the whole list from https://source.chromium.org/chromium/chromium/src/+/main:net/http/transport_security_state_static.json (The file is very large (13 MB) and may crash. Recommended viewing a raw file.)

   Discuss what you have found with your peers or tutor.