Homework 1

COM402 - Information Security and Privacy 2017

- In this homework, you will be collecting secret tokens, one for each task. Please save these tokens, as you need to submit them on Moodle as your solutions. The tokens are unique for each student. See the format submission here: http://moodle.epfl.ch/mod/assign/view.php?id=944250
- You need a Docker environment to be set up on your machine. Alternatively, you can use the provided VirtualBox image with Docker in it, as described here.
- The homework is due **Thursday, 16th of March, at 23h00 on Moodle**. Submission instructions are on Moodle.

Exercise 1: Are Clients Better Authenticated In The West Or In The East?

To discover your first token, you need to pass authentication on the DeDiS Secure Login page
http://com402.epfl.ch/hw1/ex1

You must use your personal EPFL email address as a login name. As soon as you are logged in, you will be able to see your token! The problem is that you have no clue what your password is. But maybe you can find out how the password-verification is performed and trick it? **Please do NOT insert your GASPAR password in this exercise!**

Exercise 2: Is After-Lunch Time Best for Hacking?

The Evil Corp is on track again. You do not like many things about them but the worst one is that, if you log in once, they start tracking you. Even if you leave and come back as a new person, they will recognise you. Fortunately, you have heard rumours that their authentication mechanism is severely broken, and it is not so difficult to log in as a simple user. But can you go even further and spy on the whole world?

The login web page of Evil Corp is located at
http://com402.epfl.ch/hw1/ex2/login

If you manage to spy on the whole world, you will see your secret on the web page.
Use again your personal EPFL email address as a username. And, as in the previous exercise, **using your GASPAR password here is NOT the best idea**.

Exercise 3: All their parcels are belong to you[1]

In this and the following exercise, you need to imagine that you are in a cyber-cafe. The cyber-cafe with the smell of freshly brewed coffee but also crowded with computer newbies who come here to browse the Internet and shopping websites.

You have managed to take the control over the router in the cafe and, as a result, all the traffic in the cafe goes through your machine so you can intercept and modify HTTP packets. Not bad!

You notice that your neighbour in the cafe has added some product in a shopping basket on one of the shopping websites. You find this product very cool and decide to have it delivered to you instead, by **substituting the shipping address of HTTP purchase request with your personal EPFL email address**. The other fields need to stay the same.

Steps:

a. Setup

First, you must create a new directory for this exercise (`mkdir`) and go (`cd`) to that directory. Then create an empty python script here, for example with:

```
touch interceptor.py
```

Then download the shell script `run_dockers.sh` from Moodle *to the same directory*. The script pulls and runs two Docker images, *generator* representing a machine of a random client in the cafe and *attacker* is your machine. The script will create a mapping from your current directory to /shared. You can then run the script as follows:

```
run_dockers.sh your_personal_epfl_email_address
```

The *generator* sends HTTP packets to `com402.epfl.ch/hw1/ex3/shipping` and `com402.epfl.ch/hw1/ex4/transaction`.

To make it simpler for you, *generator* probabilistically sends all the necessary information in a couple of minutes and then just repeats it again in a random order.

You can now connect to the *attacker* docker container:

```
docker exec -it attacker /bin/bash
```

[1] Inspired by https://en.wikipedia.org/wiki/All_your_base_are_belong_to_us

Since the attacker container is the router, it must do proper NAT (https://en.wikipedia.org/wiki/Network_address_translation). To enable that you can use iptables. Iptables is an user-space application program that allows a system administrator to configure the tables provided by the Linux kernel firewall.
To enable NAT, type the following:

```
docker exec attacker /bin/bash -c 'iptables -t nat -A POSTROUTING
-j MASQUERADE'
```

Since the attacker's goal is to intercept traffic, we now need a way to tell the linux kernel to give us all packets coming from the generator. To do that, we set up an iptables rule that redirect some specific packets to a special user-space queue called NFQUEUE. To enable it:

```
docker exec attacker /bin/bash -c 'iptables -A FORWARD -s
172.16.0.2 -p tcp --dport 80 -j NFQUEUE --queue-num 0'
```

- Every packet that goes through the attacker machine coming from the generator matches the iptables FORWARD chain. In our case the generator sends packets to `com402.epfl.ch`, but the packet passes through the attacker, since the attacker has control of the router.
- For theses packets, check if the source address is the one from the generator, and if it's a TCP packet with the destination port being 80 (i.e., mostly HTTP traffic).
- In that case, redirect that packet to a user-space queue: NFQUEUE number 0.

All packets in the NFQUEUE can be read by user space programs which can decide if a packet goes on or is dropped.

b. Seeing traffic

This section explains you how can you programmatically read the traffic. Please note that **all libraries needed are already installed inside the attacker container**.

1. Netfilter

In order to read from this NFQUEUE in Python3, we need to use the NetFilterQueue library. All documentation can be found at https://pypi.python.org/pypi/NetfilterQueue .

Some notes on NetfilterQueue:

- Follow the first example, DON'T bind over a socket.
- You need to bind to the queue number 0

- To not overload the queue, pass a third parameter to the bind method specifying what should the length of the queue.
  - `nfqueue.bind(0, callback, 100)`
- To get the raw bytes, call `pkt.get_payload()`
- To drop the packet: `pkg.drop()`. For this exercise, you can safely drop all packets.

You can test your script by running `python3 shared/interceptor.py` from **inside** the container or by running from your machine:

```
docker exec -it attacker python3 shared/interc eptor.py
```

2. Scapy

Now that you can see all packets within the callback function, you need to parse them. Indeed, netfilter will give you RAW packets as bytes, containing all layers from the Ethernet frame to the Application layer (See https://en.wikipedia.org/wiki/Data_link_layer for more information).

Scapy is a very powerful packet manipulation Python library. It's easy to parse packet and modify traffic with it. The official documentation is at http://www.secdev.org/projects/scapy/doc/ . A more "practical" tutorial can be found here https://bt3gl.github.io/black-hat-python-infinite-possibilities-with-the-scapy-module.html . Since the goal is not to become a Scapy expert, here's the list of methods you will need for this exercise:

- Create a IP packet object out of the raw bytes: `ip = IP(raw)`
- Check if there is a specific layer in the packet: `ip.haslayer(`*`layer`*`)`
- `Get a specific layer: ip[`*`layer`*`]`
  - `Some useful layers: TCP, Raw (application layer)`
- It's good to check if the destination port is 80 on the TCP layer:
  - `tcp.dport == 80`
  - Question: Why would there not be any application layer in a TCP packet ?
- Get the application layer payload: `http = ip[Raw].load.decode()`

3. HTTP

Now that you have the http payload, you need to parse it ! As a reminder, here's a generic HTTP traffic payload containing JSON:

```
POST /hw1/ex3/shipping HTTP/1.1
Host: com402.epfl.ch
Content-Length: 91
Content-Type: application/json
User-Agent: Dumb Generator
```

`<JSON CONTENT>`

In python, `json.loads(data)` will prove useful!

c. Send traffic

Once you successfully parsed and changed the shipping address in the HTTP payload, you can send **a new packet** containing the right json directly to the server, specifically to to `com402.epfl.ch/hw1/ex3/shipping` . In this exercise, you can use the requests library (documentation at http://docs.python-requests.org/en/master/).

Please note that there will be only **multiple packets** with a shipping address in their HTTP payload, **but they are all identical** (so that missing one shipping packet does not require to restart the generator). **Thus if you have intercepted one such packet, you do not need to search for more.**

If you do everything correctly, the shopping website at `com402.epfl.ch/hw1/ex3/shipping` will answer with an HTTP packet carrying `200` status code and a secret token in its payload.

Exercise 4: Keep Your Feet Warm And Sensitive Data Protected

You are still in the cyber-cafe, but now you realize that you can do much more than sending a product to your email address. People in the cafe are using their credit cards for payments over the Internet and passwords for logging in into their online accounts. And all this data is not encrypted! How convenient! You have to find a way to look at all the traffic and find some valuable information.

You use the same *generator* and *attacker* containers as in the previous exercise and need to search through the traffic you sniff. Your task is to find in the traffic **five distinct pieces of sensitive information, either credit card numbers or passwords**. Specifically, **some packets** contain in their HTTP payload sensitive information that has the following format:

`cc --- card_number` , where *`card_number`* is either `xxxx.xxxx.xxxx.xxxx` or `xxxx/xxxx/xxxx/xxxx` and `x` is a digit `0-9`.

or

`pwd --- password`, where *`password`* is a mix of digits `0-9`, upper-case letters `A-Z`, symbols `: ; < = > ? @` The passwords also have a length between 8 and 30 symbols. Correct passwords **DO NOT** contain lower-case letters.

Sensitive information can be preceded or succeeded by random strings containing a mix of digits `0-9`, upper-case letters `A-Z`, and lower-case letters `a-z`. The secret is always separated through a space from these random strings, one space before the sensitive word and one after. For instance, an HTTP payload containing sensitive information is:

```
lKJdeL324bhvw34FbnFa cc --- 1234.5678.9012.3456
khj35jvDKldbFGhj6fre
```

But be careful! Some packets contain strings that resemble the format of secrets, but do not match the exact format description. Thus, they do not contain valid secrets and you should not extract information from them.

As soon as you collect all five distinct secrets, you need to create a single HTTP packet with a JSON payload of the following form:

```
{
"student_email": student_EPFL_email_address ,
"secrets": [secret1, secret2, secret3, secret4, secret5]
}
```

where you give your EPFL e-mail address as in the previous exercises and *`secrets 1-5`* with the secrets you have sniffed. Afterwards, you need to send the HTTP packet you have created to `com402.epfl.ch/hw1/ex4/sensitive` .

If you have found **all** the five pieces correctly, the server will respond with an HTTP packet carrying the `200` success code and your final secret token!

Good luck!