Course: EGDF20 Diploma in Electronic and Computer Engineering

Module: EGE356 IoT System Architecture & Technology

Lab 5: Gateway Web Services and Security

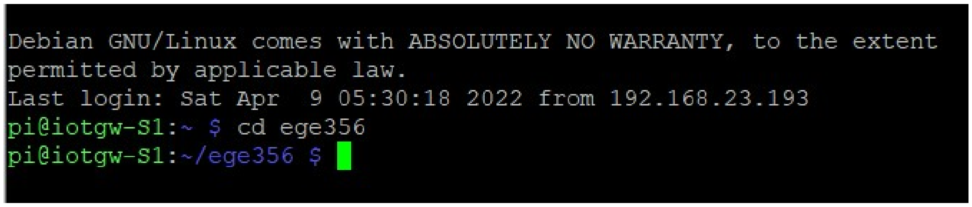
Objectives:

1. Understanding Web Services
2. Network Analysis with Wireshark
3. Web Service Token Authentication

Background: In this lab, learners will learn to understand the Web Services through network analysis using Wireshark and understand that the use of token authentication should be used with encryption (https) to secure the web application.

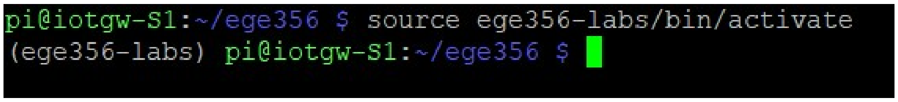
1. In this part of the lab, the files and folders used is will be similar to part 3 of Lab 4. Learners may use the steps below or refer to step 1 of Lab 4.

```
cd ege356
```



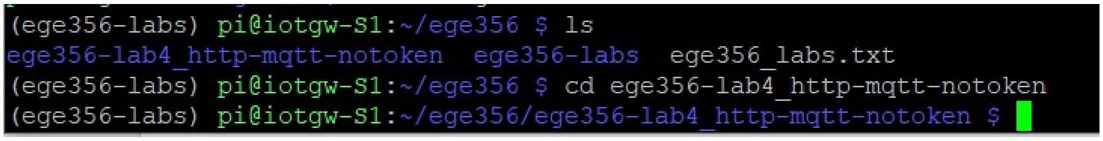
```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Apr  9 05:30:18 2022 from 192.168.23.193
pi@iotgw-S1:~ $ cd ege356
pi@iotgw-S1:~/ege356 $
```

```
source ege356-labs/bin/activate
```



```
pi@iotgw-S1:~/ege356 $ source ege356-labs/bin/activate
(ege356-labs) pi@iotgw-S1:~/ege356 $
```

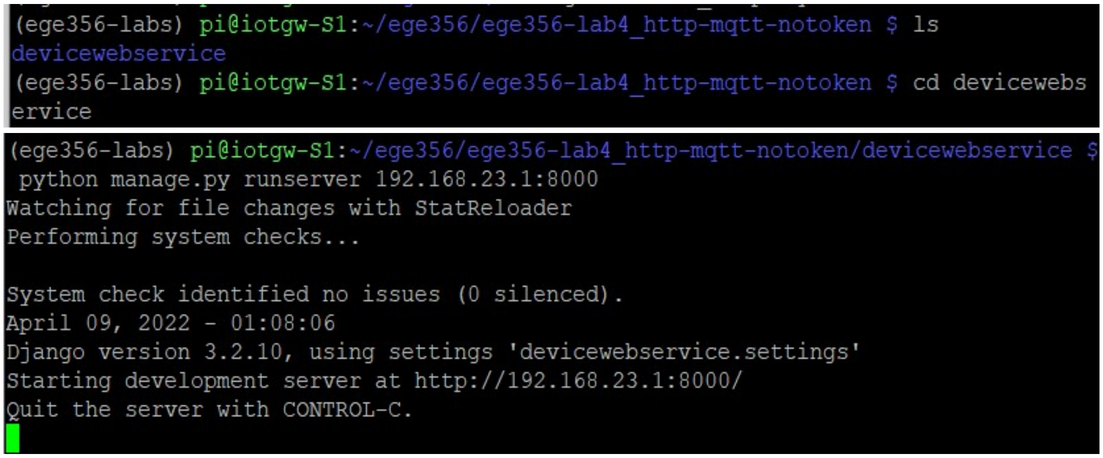
```
cd ege356-lab4_http-mqtt-notoken
```



```
(ege356-labs) pi@iotgw-S1:~/ege356 $ ls
ege356-lab4_http-mqtt-notoken  ege356-labs  ege356_labs.txt
(ege356-labs) pi@iotgw-S1:~/ege356 $ cd ege356-lab4_http-mqtt-notoken
(ege356-labs) pi@iotgw-S1:~/ege356/ege356-lab4_http-mqtt-notoken $
```

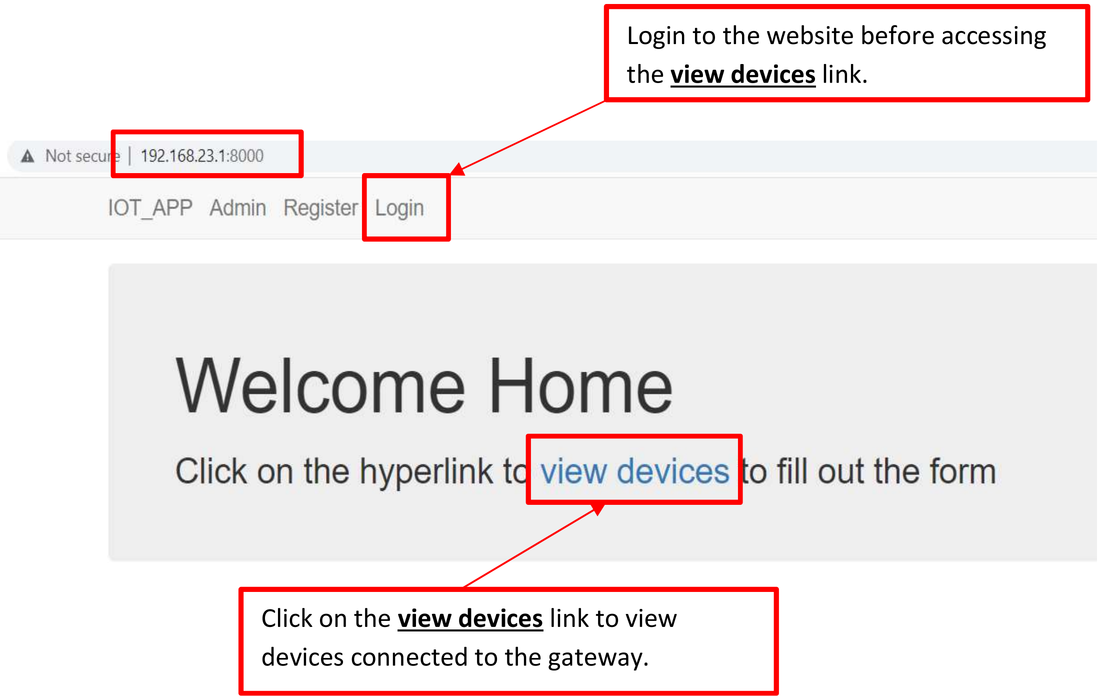
```
ls
cd devicewebservice

python manage.py runserver 192.168.X.X:8000
```

Use the gateway's IP address, e.g. 192.168.23.1



```
(ege356-labs) pi@iotgw-S1:~/ege356/ege356-lab4_http-mqtt-notoken $ ls
devicewebservice
(ege356-labs) pi@iotgw-S1:~/ege356/ege356-lab4_http-mqtt-notoken $ cd devicewebs
ervice
(ege356-labs) pi@iotgw-S1:~/ege356/ege356-lab4_http-mqtt-notoken/devicewebservice $
 python manage.py runserver 192.168.23.1:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 09, 2022 - 01:08:06
Django version 3.2.10, using settings 'devicewebservice.settings'
Starting development server at http://192.168.23.1:8000/
Quit the server with CONTROL-C.
```

2. Launch google web browser to access the webpage. To access the web server application, type in http://192.168.X.X:8000. Note that the IP address should be the same as ip address used in step 1, the gateway IP address.



Welcome Home

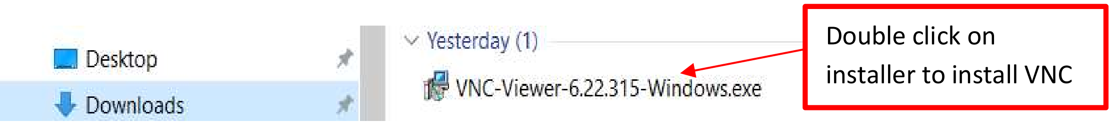
Click on the hyperlink to view devices to fill out the form

Login to the website before accessing the **view devices** link.

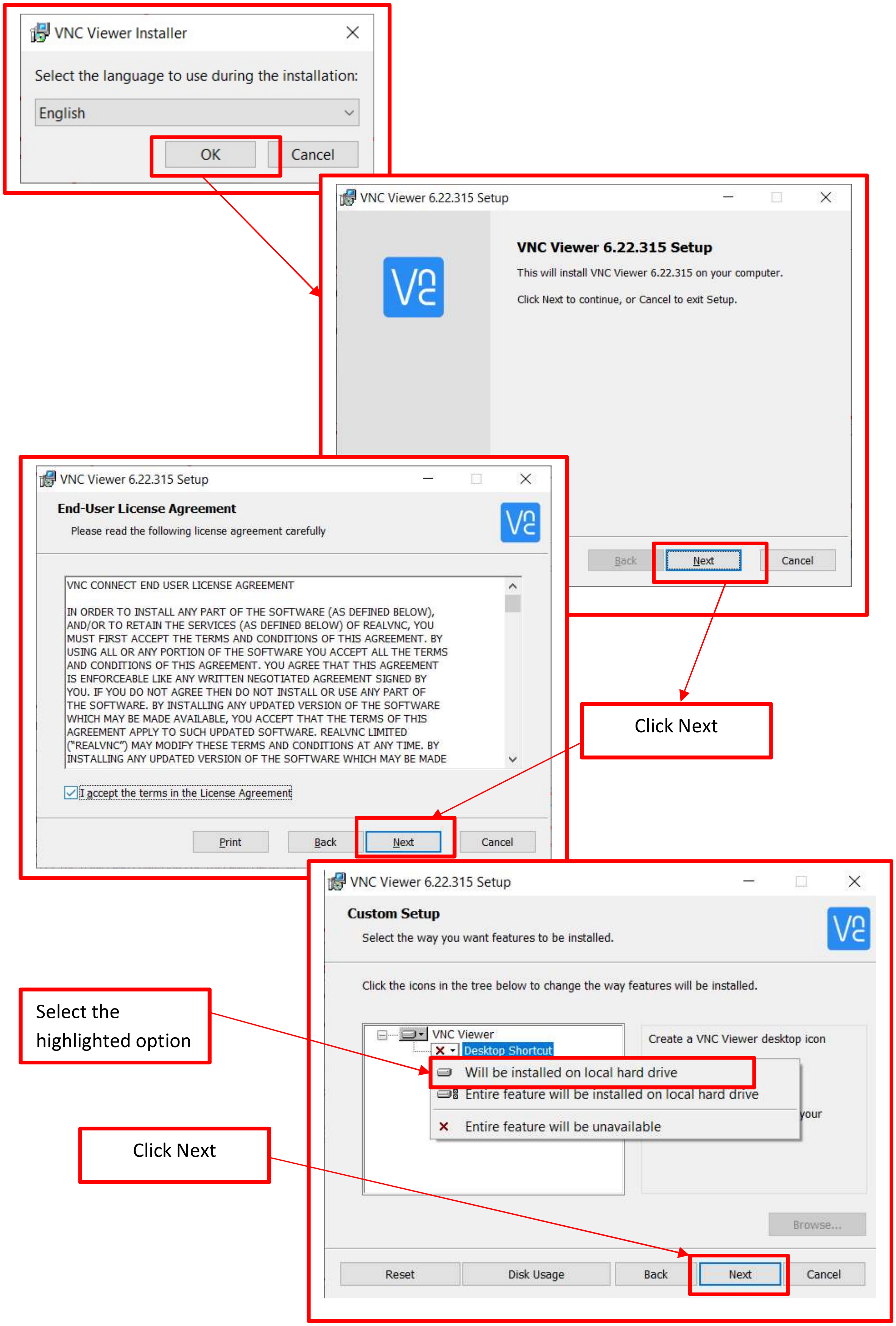
Click on the **view devices** link to view devices connected to the gateway.

3. Download and install VNC to learner’s laptop via the link provided in brightspace. Alternatively, learners may download the latest version from the weblink below:

https://www.realvnc.com/en/connect/download/viewer/



Double click on installer to install VNC



VNC Viewer Installer

Select the language to use during the installation:

English

VNC Viewer 6.22.315 Setup

VNC Viewer 6.22.315 Setup

This will install VNC Viewer 6.22.315 on your computer.

Click Next to continue, or Cancel to exit Setup.

Click Next

VNC Viewer 6.22.315 Setup

End-User License Agreement

Please read the following license agreement carefully

VNC CONNECT END USER LICENSE AGREEMENT

IN ORDER TO INSTALL ANY PART OF THE SOFTWARE (AS DEFINED BELOW), AND/OR TO RETAIN THE SERVICES (AS DEFINED BELOW) OF REALVNC, YOU MUST FIRST ACCEPT THE TERMS AND CONDITIONS OF THIS AGREEMENT. BY USING ALL OR ANY PORTION OF THE SOFTWARE YOU ACCEPT ALL THE TERMS AND CONDITIONS OF THIS AGREEMENT. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU. IF YOU DO NOT AGREE THEN DO NOT INSTALL OR USE ANY PART OF THE SOFTWARE. BY INSTALLING ANY UPDATED VERSION OF THE SOFTWARE WHICH MAY BE MADE AVAILABLE, YOU ACCEPT THAT THE TERMS OF THIS AGREEMENT APPLY TO SUCH UPDATED SOFTWARE. REALVNC LIMITED ("REALVNC") MAY MODIFY THESE TERMS AND CONDITIONS AT ANY TIME. BY INSTALLING ANY UPDATED VERSION OF THE SOFTWARE WHICH MAY BE MADE

☑ I accept the terms in the License Agreement

VNC Viewer 6.22.315 Setup

Custom Setup

Select the way you want features to be installed.

Click the icons in the tree below to change the way features will be installed.

- VNC Viewer
  - Desktop Shortcut

Will be installed on local hard drive

Entire feature will be installed on local hard drive

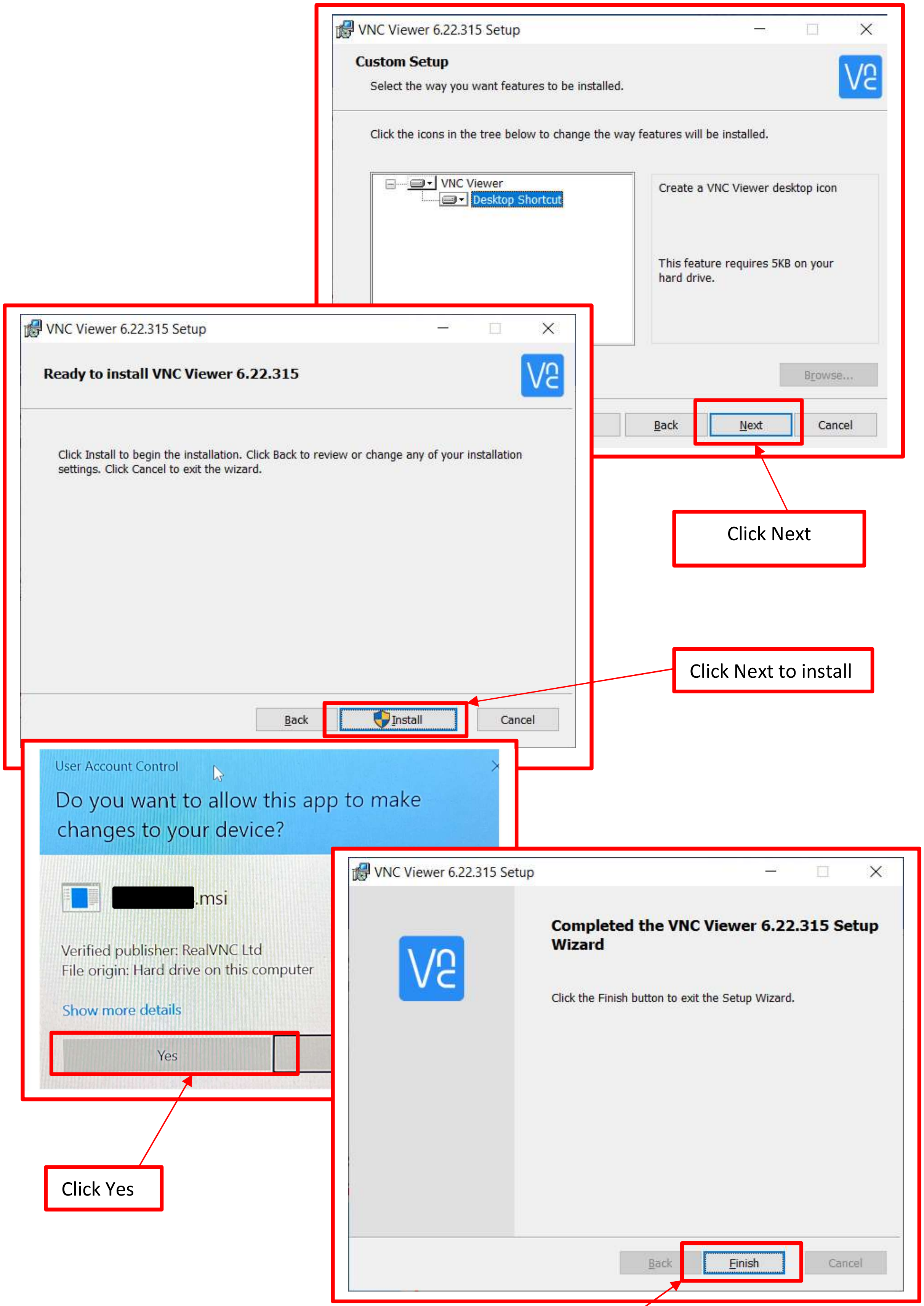
Entire feature will be unavailable

Create a VNC Viewer desktop icon

your

Select the highlighted option

Click Next



VNC Viewer 6.22.315 Setup

Custom Setup

Select the way you want features to be installed.

Click the icons in the tree below to change the way features will be installed.

Create a VNC Viewer desktop icon

This feature requires 5KB on your hard drive.

Click Next

VNC Viewer 6.22.315 Setup

Ready to install VNC Viewer 6.22.315

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

Click Next to install

User Account Control

Do you want to allow this app to make changes to your device?

[redacted].msi

Verified publisher: RealVNC Ltd
File origin: Hard drive on this computer

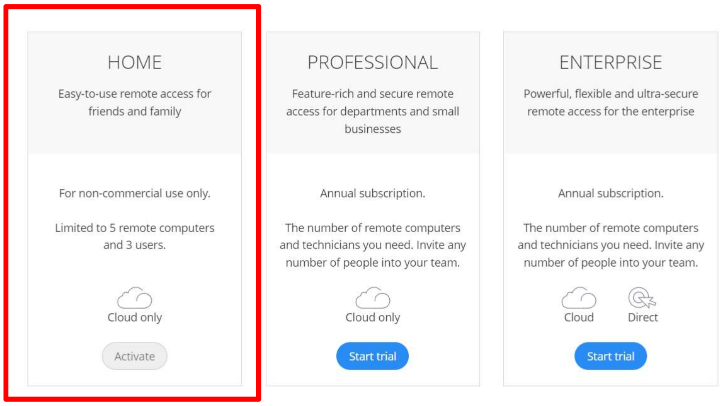
Click Yes

VNC Viewer 6.22.315 Setup

Completed the VNC Viewer 6.22.315 Setup Wizard

Click the Finish button to exit the Setup Wizard.

Click on Finish to complete the installation.

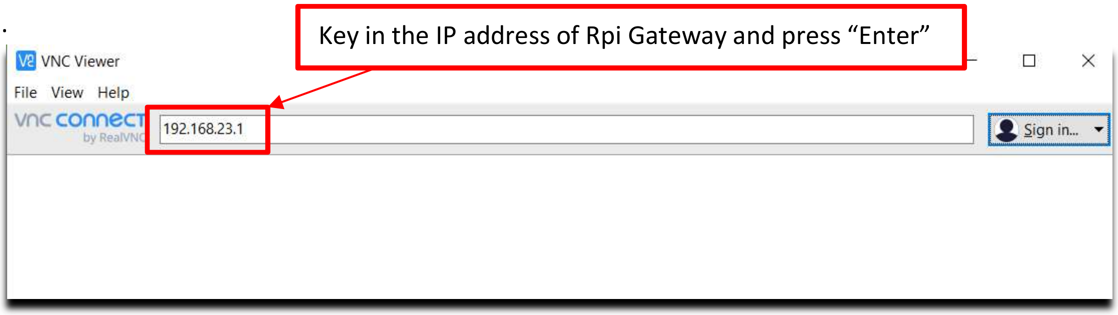

HOME
Easy-to-use remote access for friends and family
For non-commercial use only.
Limited to 5 remote computers and 3 users.
Cloud only
Activate
PROFESSIONAL
Feature-rich and secure remote access for departments and small businesses
Annual subscription.
The number of remote computers and technicians you need. Invite any number of people into your team.
Cloud only
Start trial
ENTERPRISE
Powerful, flexible and ultra-secure remote access for the enterprise
Annual subscription.
The number of remote computers and technicians you need. Invite any number of people into your team.
Cloud
Direct
Start trial

Online registration is not compulsory but to register, select the Home edition for personal use.

4. In this part of the lab, learners will be accessing RPi through VNC so as to use wireshark to analyse network packet data.

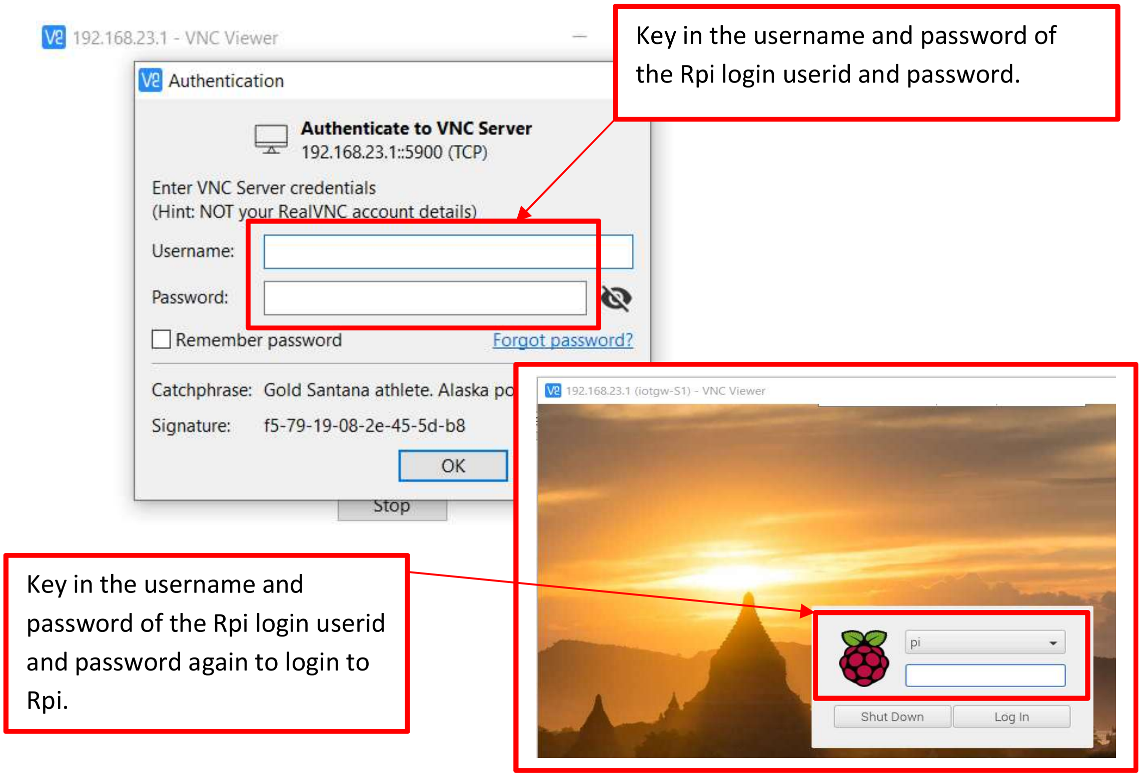
5. Launch VNC as shown below and key in the IP Address of the Rpi gateway to access the Rpi desktop as shown below.

6.



192.168.23.1

Key in the IP address of Rpi Gateway and press “Enter”


192.168.23.1 - VNC Viewer
Authentication
Stop
192.168.23.1 (iotgw-S1) - VNC Viewer
Shut Down
Log In

Authenticate to VNC Server
192.168.23.1::5900 (TCP)

Enter VNC Server credentials
(Hint: NOT your RealVNC account details)

Username:

Password:

□ Remember password Forgot password?

Catchphrase: Gold Santana athlete. Alaska po

Signature: f5-79-19-08-2e-45-5d-b8

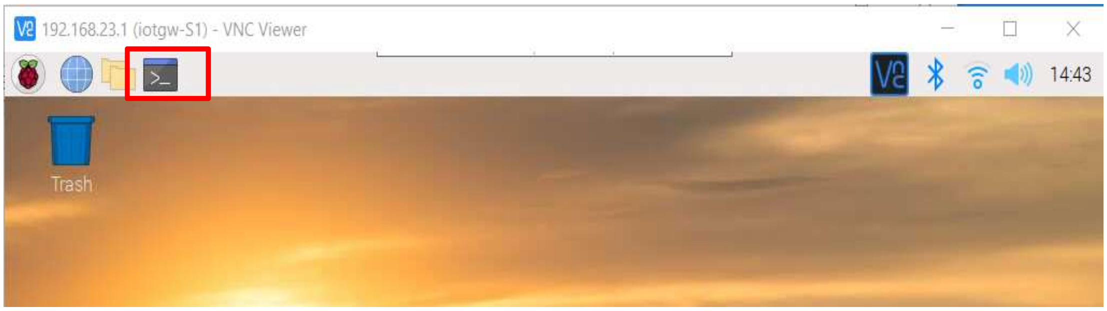
OK

pi

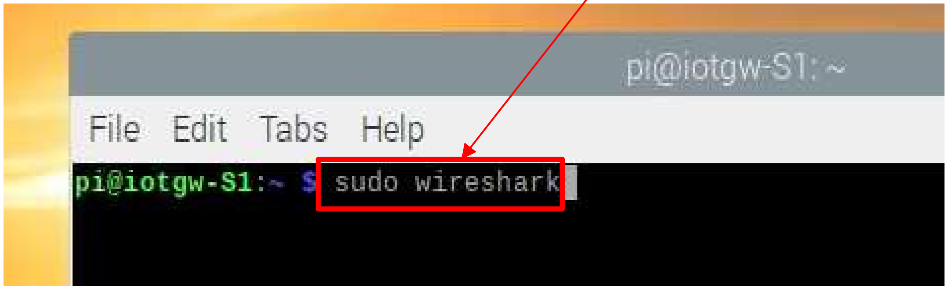
Key in the username and password of the Rpi login userid and password.

Key in the username and password of the Rpi login userid and password again to login to Rpi.

7. In the Rpi, launch the terminal app as shown below:

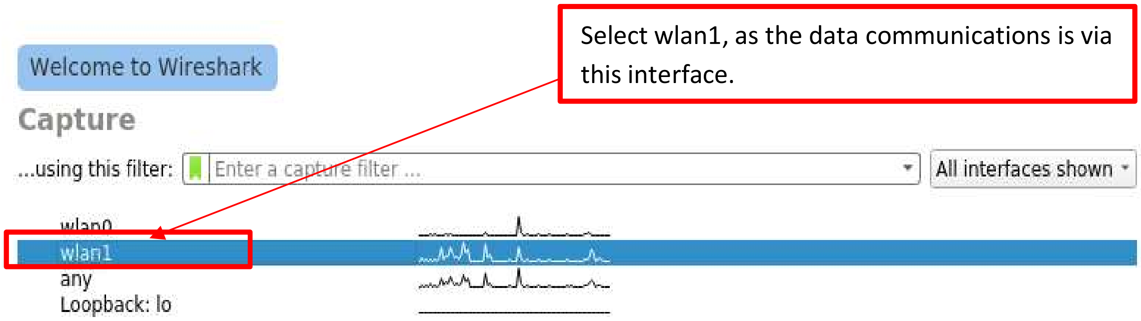


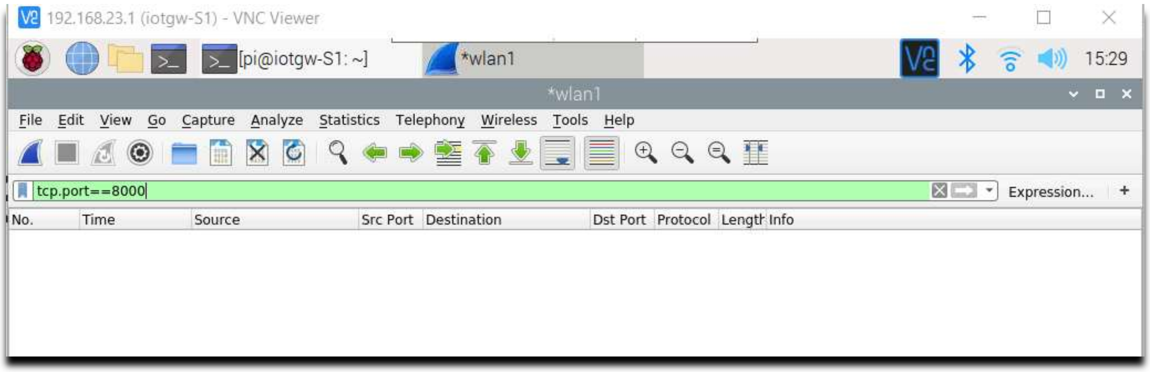
Launch wireshark with admin rights



```
pi@iotgw-S1:~ $ sudo wireshark
```

Select wlan1, as the data communications is via this interface.

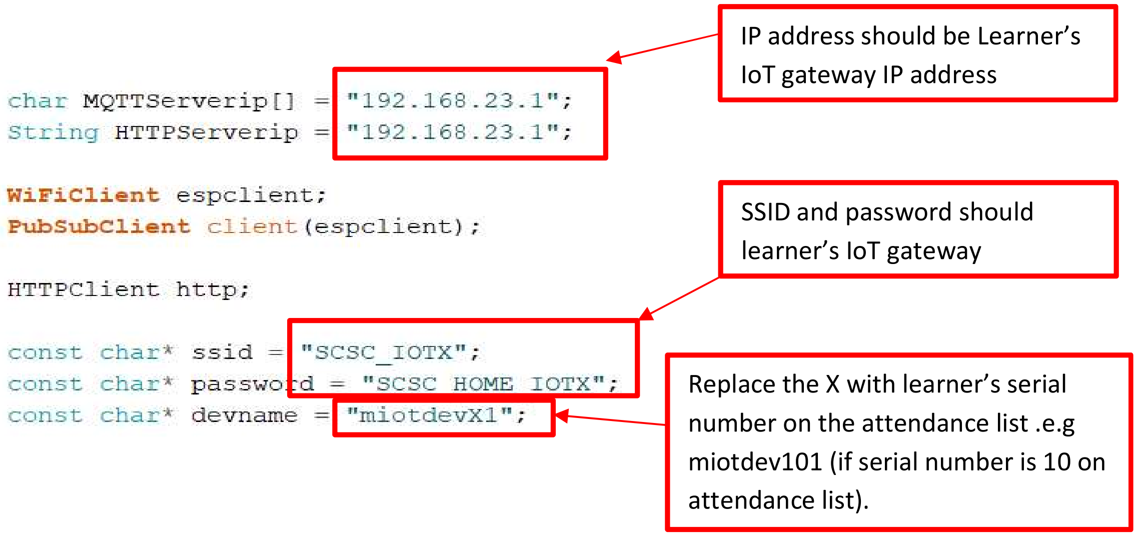




```
tcp.port==8000
```

8. Type tcp.port == 8000 in the display filter so as to see data traffic related to tcp port 8000, which is the data traffic of interest. The data to be observed will be from M5StickC device to Rpi gateway or from learner’s laptop to the Rpi gateway. ***Do not start the network capture yet***.

9. Download the Arduino zipped folder ege356_lab5_wifi_http_mqtt.zip from Brightspace and extract the contents to the Arduino folder (***ege356_lab5_wifi_http_mqtt.ino***). Make the following changes before loading the code to the M5StickC device.



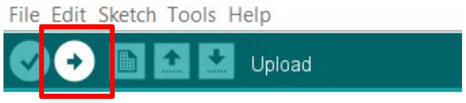
```
char MQTTServerip[]  = "192.168.23.1";
String HTTPServerip  = "192.168.23.1";

WiFiClient espclient;
PubSubClient client(espclient);

HTTPClient http;

const char* ssid = "SCSC_IOTX";
const char* password = "SCSC_HOME_IOTX";
const char* devname = "miotdevX1";
```

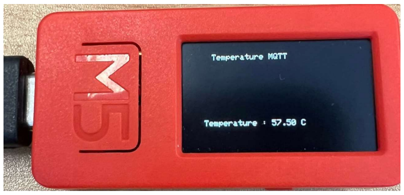
IP address should be Learner's IoT gateway IP address

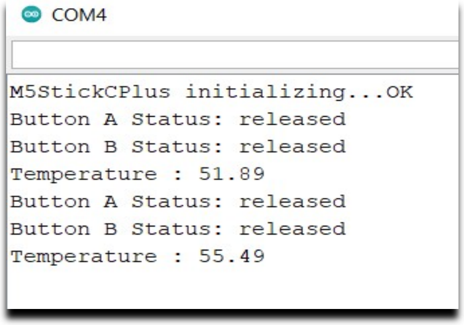
SSID and password should learner's IoT gateway

Replace the X with learner's serial number on the attendance list .e.g miotdev101 (if serial number is 10 on attendance list).

10. Connect the M5StickC Plus device to the laptop/Lab PC and load the code to the M5StickC plus device.

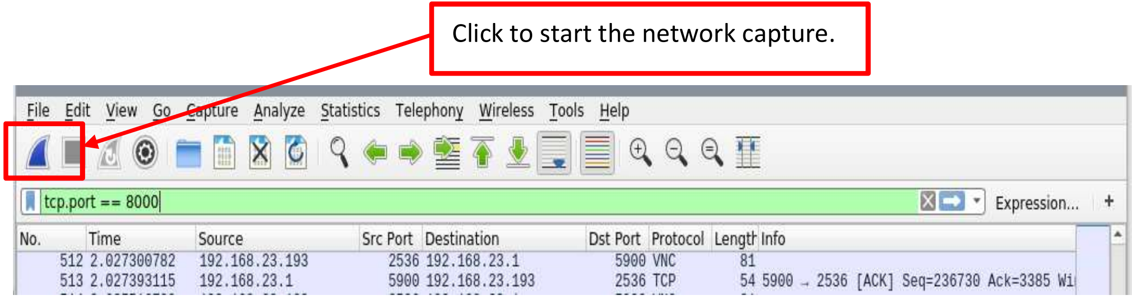






```
M5StickCPlus initializing...OK
Button A Status: released
Button B Status: released
Temperature : 51.89
Button A Status: released
Button B Status: released
Temperature : 55.49
```

11. Start the network capture.



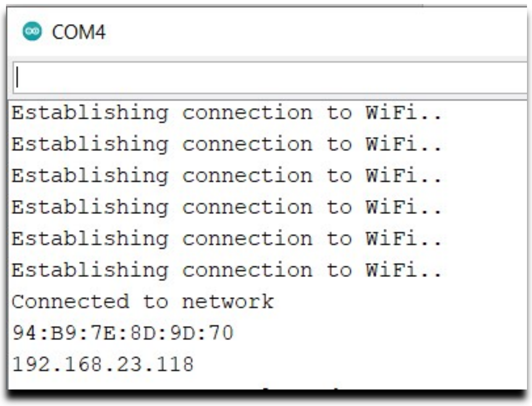
Click to start the network capture.

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...
+

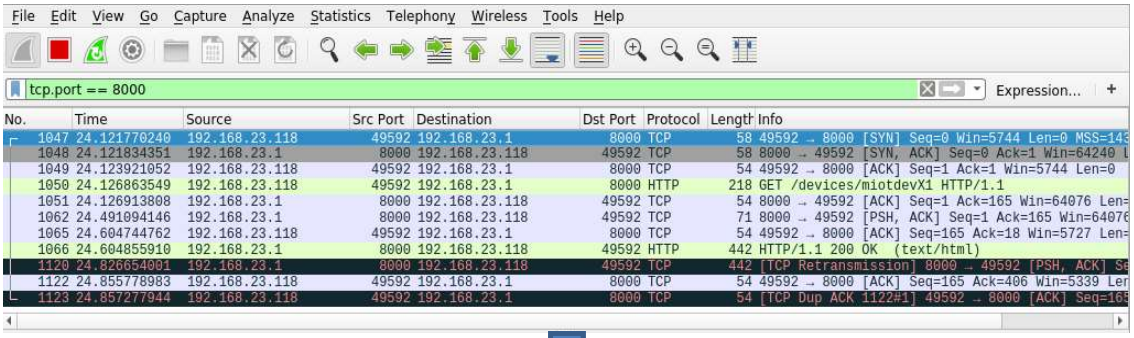
tcp.port == 8000

```
No.   Time         Source          Src Port Destination    Dst Port Protocol Length Info
  512 2.027300782  192.168.23.193  2536 192.168.23.1       5900 VNC     81
  513 2.027393115  192.168.23.1    5900 192.168.23.193     2536 TCP     54 5900 → 2536 [ACK] Seq=236730 Ack=3385 Wi
```

Once the network capture has started, click on M5StickC Button A. The M5StickC device will connect to the wifi, and send a http get request to Gateway Webservice. From the network packet capture, observe the message sent to the Gateway.



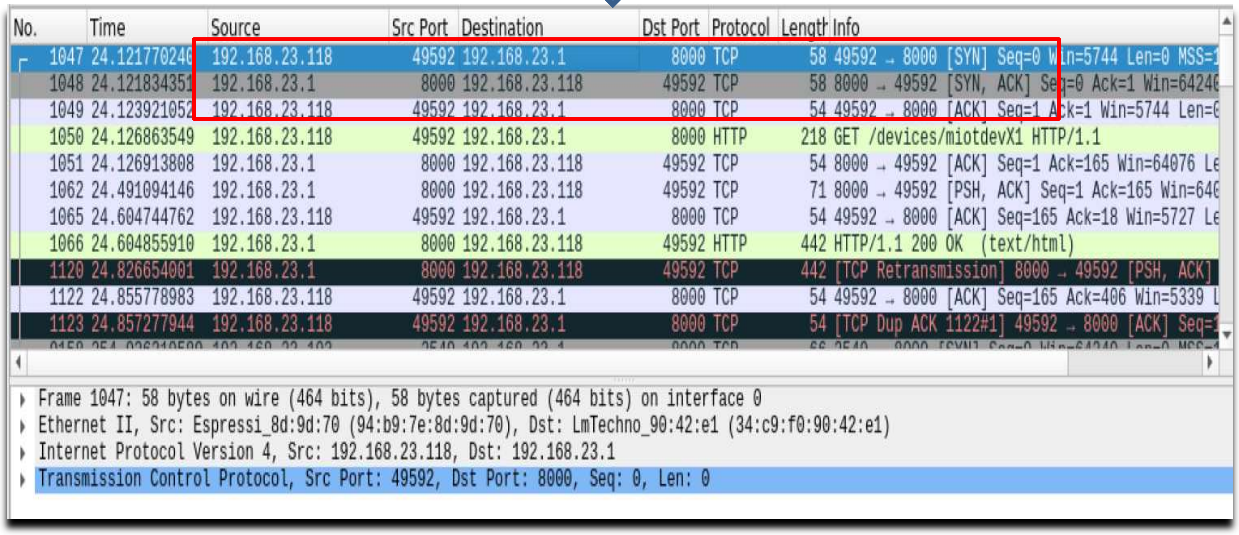
```
Establishing connection to WiFi..
Establishing connection to WiFi..
Establishing connection to WiFi..
Establishing connection to WiFi..
Establishing connection to WiFi..
Establishing connection to WiFi..
Connected to network
94:B9:7E:8D:9D:70
192.168.23.118
```



File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...
+

tcp.port == 8000

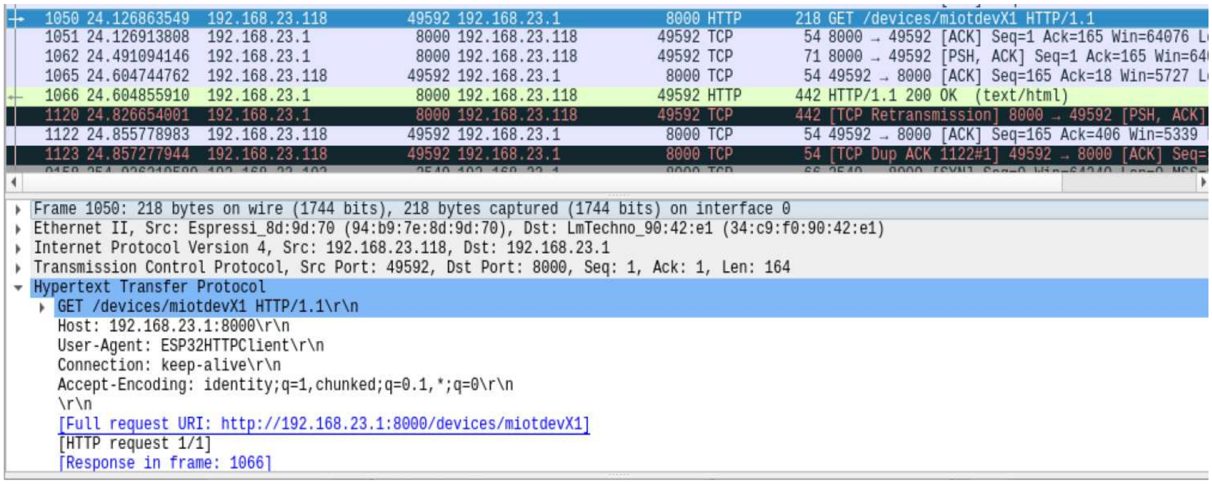
```
No.   Time          Source          Src Port Destination     Dst Port Protocol Length Info
1047 24.121770240  192.168.23.118  49592 192.168.23.1    8000 TCP    58 49592 → 8000 [SYN] Seq=0 Win=5744 Len=0 MSS=143
1048 24.121834351  192.168.23.1     8000 192.168.23.118 49592 TCP    58 8000 → 49592 [SYN, ACK] Seq=0 Ack=1 Win=64240 L
1049 24.123921052  192.168.23.118  49592 192.168.23.1    8000 TCP    54 49592 → 8000 [ACK] Seq=1 Ack=1 Win=5744 Len=0
1050 24.126863549  192.168.23.118  49592 192.168.23.1    8000 HTTP  218 GET /devices/miotdevX1 HTTP/1.1
1051 24.126913808  192.168.23.1     8000 192.168.23.118 49592 TCP    54 8000 → 49592 [ACK] Seq=1 Ack=165 Win=64076 Len=
1062 24.491094146  192.168.23.1     8000 192.168.23.118 49592 TCP    71 8000 → 49592 [PSH, ACK] Seq=1 Ack=165 Win=64076
1065 24.604744762  192.168.23.118  49592 192.168.23.1    8000 TCP    54 49592 → 8000 [ACK] Seq=165 Ack=18 Win=5727 Len=
1066 24.604855910  192.168.23.1     8000 192.168.23.118 49592 HTTP  442 HTTP/1.1 200 OK  (text/html)
1120 24.826654001  192.168.23.1     8000 192.168.23.118 49592 TCP   442 [TCP Retransmission] 8000 → 49592 [PSH, ACK] Se
1122 24.855778983  192.168.23.118  49592 192.168.23.1    8000 TCP    54 49592 → 8000 [ACK] Seq=165 Ack=406 Win=5339 Ler
1123 24.857277944  192.168.23.118  49592 192.168.23.1    8000 TCP    54 [TCP Dup ACK 1122#1] 49592 → 8000 [ACK] Seq=165
```



```
No.   Time         Source          Src Port Destination     Dst Port Protocol Length Info
1047 24.12177024  192.168.23.118  49592 192.168.23.1    8000 TCP    58 49592 → 8000 [SYN] Seq=0 Win=5744 Len=0 MSS=1
1048 24.12183435  192.168.23.1     8000 192.168.23.118 49592 TCP    58 8000 → 49592 [SYN, ACK] Seq=0 Ack=1 Win=6424
1049 24.12392105  192.168.23.118  49592 192.168.23.1    8000 TCP    54 49592 → 8000 [ACK] Seq=1 Ack=1 Win=5744 Len=
1050 24.126863549 192.168.23.118  49592 192.168.23.1    8000 HTTP  218 GET /devices/miotdevX1 HTTP/1.1
1051 24.126913808 192.168.23.1     8000 192.168.23.118 49592 TCP    54 8000 → 49592 [ACK] Seq=1 Ack=165 Win=64076 Le
1062 24.491094146 192.168.23.1     8000 192.168.23.118 49592 TCP    71 8000 → 49592 [PSH, ACK] Seq=1 Ack=165 Win=640
1065 24.604744762 192.168.23.118  49592 192.168.23.1    8000 TCP    54 49592 → 8000 [ACK] Seq=165 Ack=18 Win=5727 Le
1066 24.604855910 192.168.23.1     8000 192.168.23.118 49592 HTTP  442 HTTP/1.1 200 OK  (text/html)
1120 24.826654001 192.168.23.1     8000 192.168.23.118 49592 TCP   442 [TCP Retransmission] 8000 → 49592 [PSH, ACK]
1122 24.855778983 192.168.23.118  49592 192.168.23.1    8000 TCP    54 49592 → 8000 [ACK] Seq=165 Ack=406 Win=5339 L
1123 24.857277944 192.168.23.118  49592 192.168.23.1    8000 TCP    54 [TCP Dup ACK 1122#1] 49592 → 8000 [ACK] Seq=

▸ Frame 1047: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface 0
▸ Ethernet II, Src: Espressi_8d:9d:70 (94:b9:7e:8d:9d:70), Dst: LmTechno_90:42:e1 (34:c9:f0:90:42:e1)
▸ Internet Protocol Version 4, Src: 192.168.23.118, Dst: 192.168.23.1
▸ Transmission Control Protocol, Src Port: 49592, Dst Port: 8000, Seq: 0, Len: 0
```

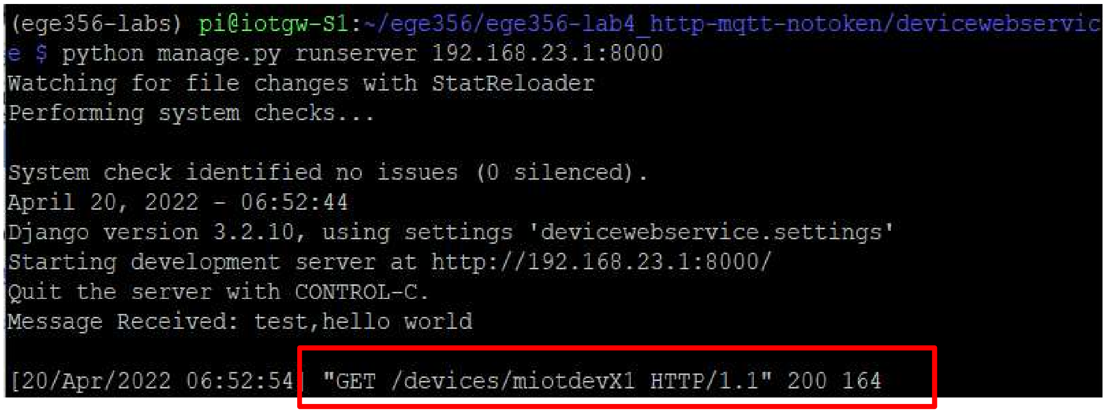
Note the TCP 3-way handshake as highlighted above.

Once the handshake has been completed and the connection established, the HTTP GET request is sent to the WebServer.



```
1050 24.126863549  192.168.23.118      49592 192.168.23.1        8000 HTTP    218 GET /devices/miotdevX1 HTTP/1.1
1051 24.126913808  192.168.23.1         8000 192.168.23.118     49592 TCP      54 8000 → 49592 [ACK] Seq=1 Ack=165 Win=64076 L
1062 24.491094146  192.168.23.1         8000 192.168.23.118     49592 TCP      71 8000 → 49592 [PSH, ACK] Seq=1 Ack=165 Win=64
1065 24.604744762  192.168.23.118      49592 192.168.23.1        8000 TCP      54 49592 → 8000 [ACK] Seq=165 Ack=18 Win=5727 L
1066 24.604855910  192.168.23.1         8000 192.168.23.118     49592 HTTP    442 HTTP/1.1 200 OK  (text/html)
1120 24.826654001  192.168.23.1         8000 192.168.23.118     49592 TCP     442 [TCP Retransmission] 8000 → 49592 [PSH, ACK]
1122 24.855778983  192.168.23.118      49592 192.168.23.1        8000 TCP      54 49592 → 8000 [ACK] Seq=165 Ack=406 Win=5339
1123 24.857277944  192.168.23.118      49592 192.168.23.1        8000 TCP      54 [TCP Dup ACK 1122#1] 49592 → 8000 [ACK] Seq=

▸ Frame 1050: 218 bytes on wire (1744 bits), 218 bytes captured (1744 bits) on interface 0
▸ Ethernet II, Src: Espressi_8d:9d:70 (94:b9:7e:8d:9d:70), Dst: LmTechno_90:42:e1 (34:c9:f0:90:42:e1)
▸ Internet Protocol Version 4, Src: 192.168.23.118, Dst: 192.168.23.1
▸ Transmission Control Protocol, Src Port: 49592, Dst Port: 8000, Seq: 1, Ack: 1, Len: 164
▾ Hypertext Transfer Protocol
  ▸ GET /devices/miotdevX1 HTTP/1.1\r\n
    Host: 192.168.23.1:8000\r\n
    User-Agent: ESP32HTTPClient\r\n
    Connection: keep-alive\r\n
    Accept-Encoding: identity;q=1,chunked;q=0.1,*;q=0\r\n
    \r\n
    [Full request URI: http://192.168.23.1:8000/devices/miotdevX1]
    [HTTP request 1/1]
    [Response in frame: 1066]
```

Note the request sent to the WebService as seen from Rpi console display.



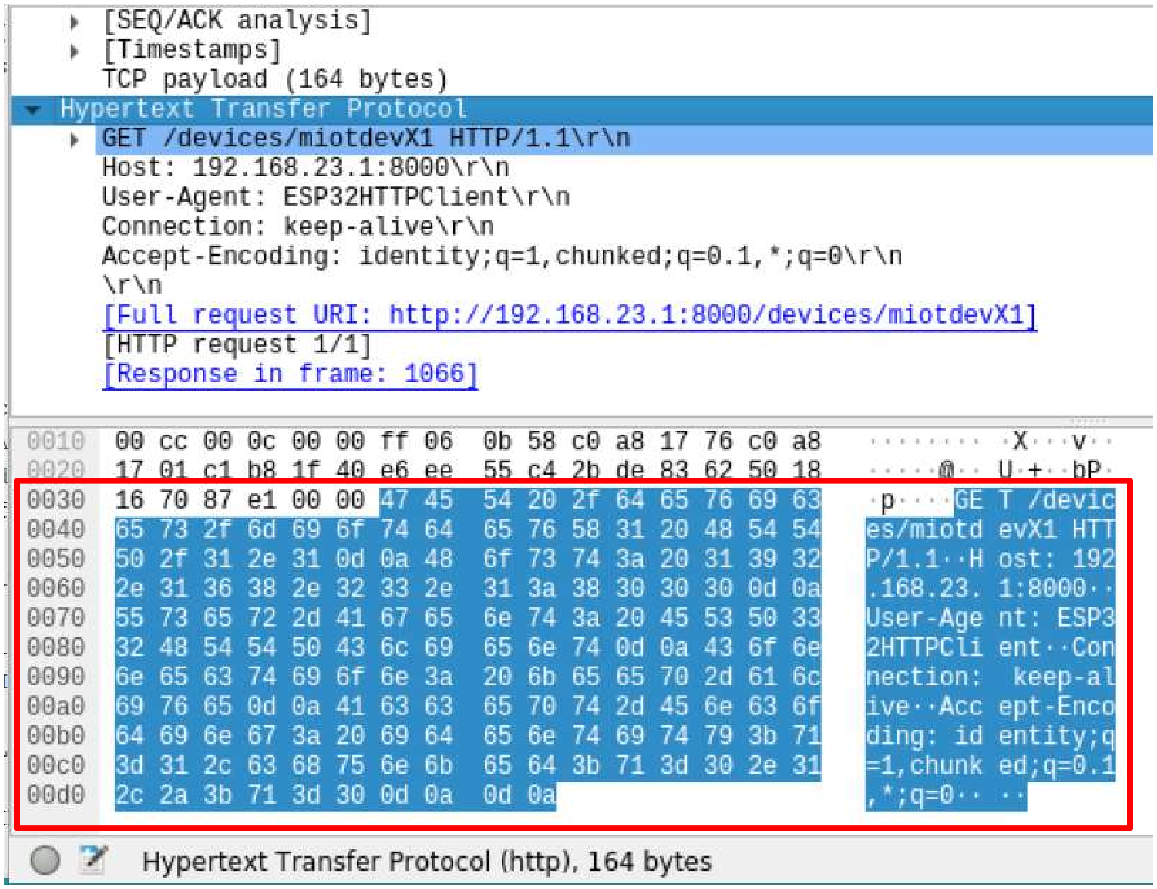
```
(ege356-labs) pi@iotgw-S1:~/ege356/ege356-lab4_http-mqtt-notoken/devicewebservic
e $ python manage.py runserver 192.168.23.1:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 20, 2022 - 06:52:44
Django version 3.2.10, using settings 'devicewebservice.settings'
Starting development server at http://192.168.23.1:8000/
Quit the server with CONTROL-C.
Message Received: test,hello world

[20/Apr/2022 06:52:54] "GET /devices/miotdevX1 HTTP/1.1" 200 164
```

What is the request method sent to the Gateway WebService?

What is the request URI sent to the WebService?

What is the full URI required to access the WebService?

The http payload is 164 bytes. From the packet bytes data pane as highlighted, provide the required calculation to obtain 164 bytes.

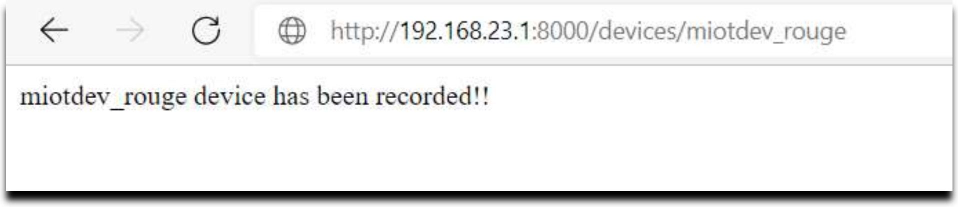


```
    ▸ [SEQ/ACK analysis]
    ▸ [Timestamps]
      TCP payload (164 bytes)
▾ Hypertext Transfer Protocol
    ▸ GET /devices/miotdevX1 HTTP/1.1\r\n
      Host: 192.168.23.1:8000\r\n
      User-Agent: ESP32HTTPClient\r\n
      Connection: keep-alive\r\n
      Accept-Encoding: identity;q=1,chunked;q=0.1,*;q=0\r\n
      \r\n
      [Full request URI: http://192.168.23.1:8000/devices/miotdevX1]
      [HTTP request 1/1]
      [Response in frame: 1066]

0010  00 cc 00 0c 00 00 ff 06  0b 58 c0 a8 17 76 c0 a8   ·········X···v··
0020  17 01 c1 b8 1f 40 e6 ee  55 c4 2b de 83 62 50 18   ·····@··U·+··bP·
0030  16 70 87 e1 00 00 47 45  54 20 2f 64 65 76 69 63   ·p····GE T /devic
0040  65 73 2f 6d 69 6f 74 64  65 76 58 31 20 48 54 54   es/miotd evX1 HTT
0050  50 2f 31 2e 31 0d 0a 48  6f 73 74 3a 20 31 39 32   P/1.1··H ost: 192
0060  2e 31 36 38 2e 32 33 2e  31 3a 38 30 30 30 0d 0a   .168.23. 1:8000··
0070  55 73 65 72 2d 41 67 65  6e 74 3a 20 45 53 50 33   User-Age nt: ESP3
0080  32 48 54 54 50 43 6c 69  65 6e 74 0d 0a 43 6f 6e   2HTTPCli ent··Con
0090  6e 65 63 74 69 6f 6e 3a  20 6b 65 65 70 2d 61 6c   nection:  keep-al
00a0  69 76 65 0d 0a 41 63 63  65 70 74 2d 45 6e 63 6f   ive··Acc ept-Enco
00b0  64 69 6e 67 3a 20 69 64  65 6e 74 69 74 79 3b 71   ding: id entity;q
00c0  3d 31 2c 63 68 75 6e 6b  65 64 3b 71 3d 30 2e 31   =1,chunk ed;q=0.1
00d0  2c 2a 3b 71 3d 30 0d 0a  0d 0a                     ,*;q=0·· ··

Hypertext Transfer Protocol (http), 164 bytes
```

Show the calculation required here:

12. Go to the browser, click on + to add a new tab, and key in the following URI:

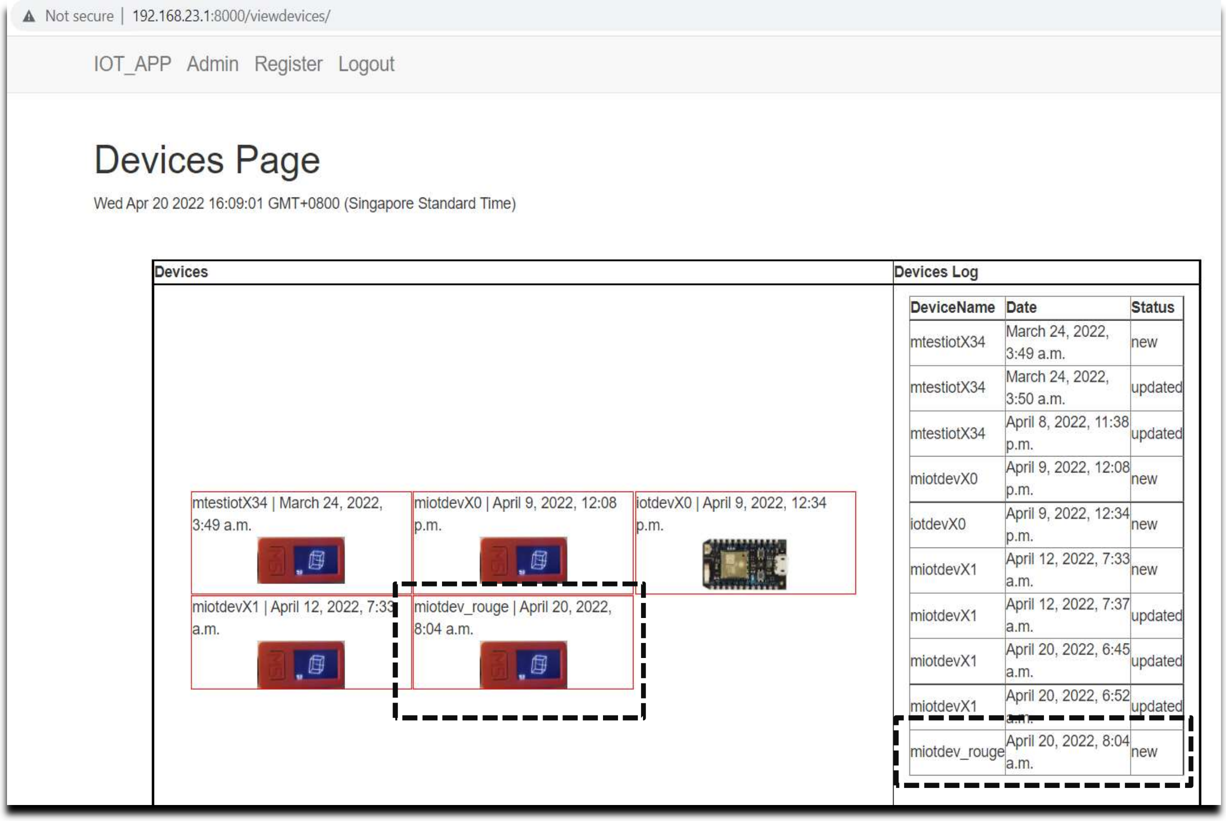
    http://192.168.X.X:8000/devices/miotdev_rouge



http://192.168.23.1:8000/devices/miotdev_rouge

miotdev_rouge device has been recorded!!

Observe the response as above from the Webservice. Return to the previous tab and observe that a new device has been added to the system. The **Devices Log** displays the time of addition of the new device. ***This is a potential security issue as anyone who managed to obtain the full URI to the Web Application would be able to register and add rouge devices to the Gateway.***



Not secure | 192.168.23.1:8000/viewdevices/
IOT_APP Admin Register Logout

Devices Page

Wed Apr 20 2022 16:09:01 GMT+0800 (Singapore Standard Time)

Devices

mtestiotX34 | March 24, 2022, 3:49 a.m.

miotdevX0 | April 9, 2022, 12:08 p.m.

iotdevX0 | April 9, 2022, 12:34 p.m.

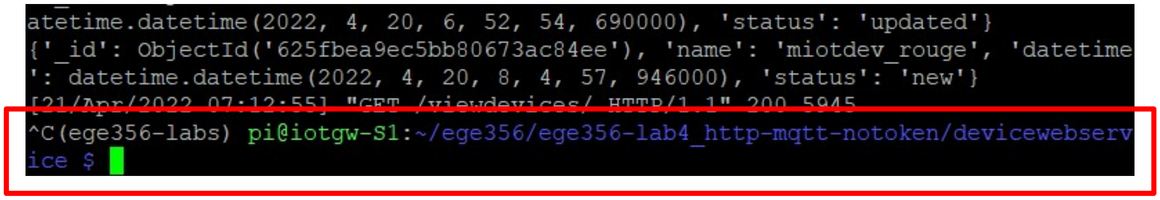
miotdevX1 | April 12, 2022, 7:33 a.m.

miotdev_rouge | April 20, 2022, 8:04 a.m.

Devices Log

DeviceName	Date	Status
mtestiotX34	March 24, 2022, 3:49 a.m.	new
mtestiotX34	March 24, 2022, 3:50 a.m.	updated
mtestiotX34	April 8, 2022, 11:38 p.m.	updated
miotdevX0	April 9, 2022, 12:08 p.m.	new
iotdevX0	April 9, 2022, 12:34 p.m.	new
miotdevX1	April 12, 2022, 7:33 a.m.	new
miotdevX1	April 12, 2022, 7:37 a.m.	updated
miotdevX1	April 20, 2022, 6:45 a.m.	updated
miotdevX1	April 20, 2022, 6:52 a.m.	updated
miotdev_rouge	April 20, 2022, 8:04 a.m.	new

13. Access the putty console window, type Ctrl-C to stop the Django Web Application.

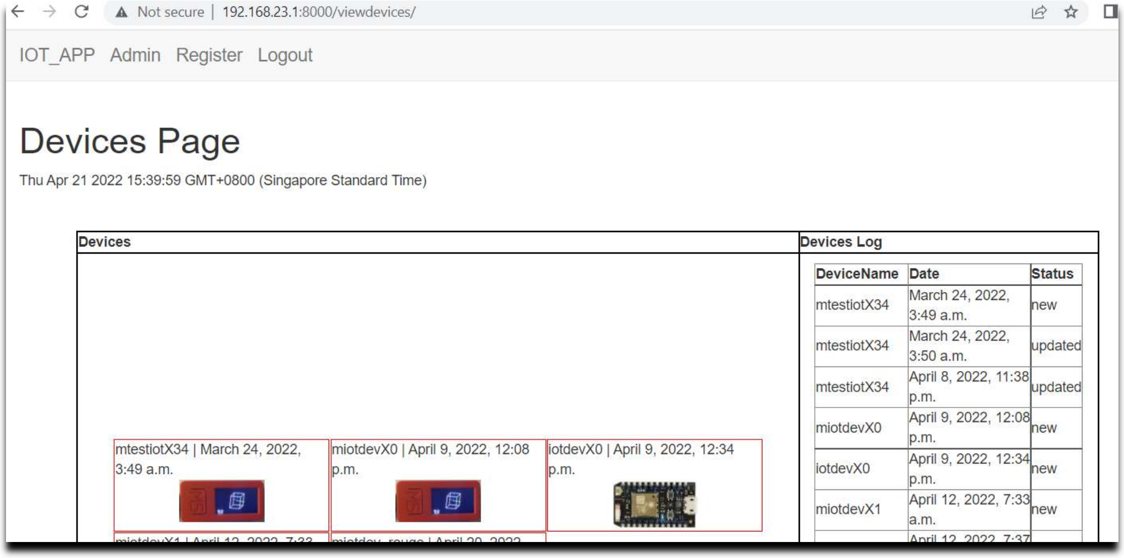


```
atetime.datetime(2022, 4, 20, 6, 52, 54, 690000), 'status': 'updated'}
{'_id': ObjectId('625fbea9ec5bb80673ac84ee'), 'name': 'miotdev_rouge', 'datetime
': datetime.datetime(2022, 4, 20, 8, 4, 57, 946000), 'status': 'new'}
[21/Apr/2022 07:12:55] "GET /viewdevices/ HTTP/1.1" 200 5945
^C(ege356-labs) pi@iotgw-S1:~/ege356/ege356-lab4_http-mqtt-notoken/devicewebserv
ice $
```

14. Type the following commands to run another Django Web Application requiring API_token for devices to be able to register to the webservice.

```
cd ../../ege356-lab4_http-mqtt-token

cd devicewebservice

python manage.py runserver 192.168.X.X:8000
```

Check that the webpage is available as shown below:



192.168.23.1:8000/viewdevices/

IOT_APP Admin Register Logout

Devices Page

Thu Apr 21 2022 15:39:59 GMT+0800 (Singapore Standard Time)

Devices

mtestiotX34 | March 24, 2022, 3:49 a.m.

miotdevX0 | April 9, 2022, 12:08 p.m.

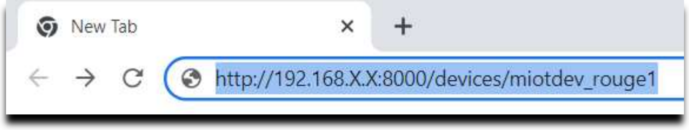
iotdevX0 | April 9, 2022, 12:34 p.m.

Devices Log

DeviceName	Date	Status
mtestiotX34	March 24, 2022, 3:49 a.m.	new
mtestiotX34	March 24, 2022, 3:50 a.m.	updated
mtestiotX34	April 8, 2022, 11:38 p.m.	updated
miotdevX0	April 9, 2022, 12:08 p.m.	new
iotdevX0	April 9, 2022, 12:34 p.m.	new
miotdevX1	April 12, 2022, 7:33 a.m.	new
	April 12, 2022, 7:37	

Key the following URI as shown below into the browser (as previously done in step 12)

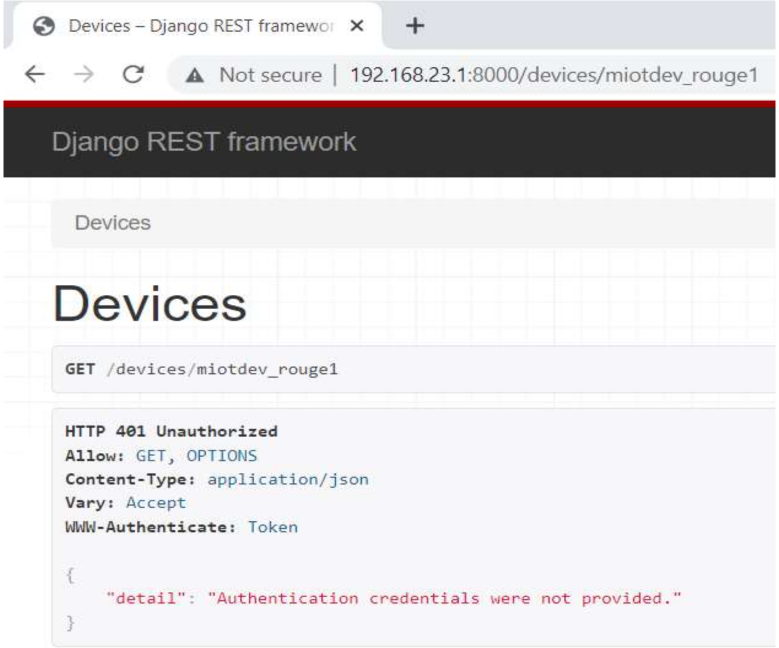
http://192.168.X.X:8000/devices/miotdev_rouge1



New Tab

http://192.168.X.X:8000/devices/miotdev_rouge1

Observe that the response from the Web App is as shown below:



Devices – Django REST framewor

192.168.23.1:8000/devices/miotdev_rouge1

Django REST framework

Devices

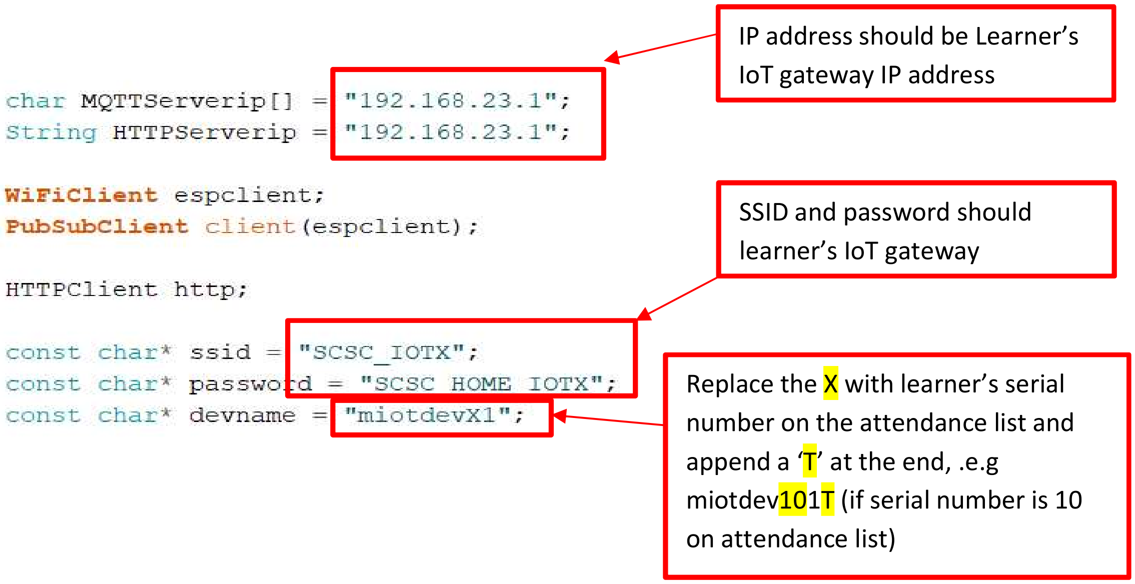
Devices

```
GET /devices/miotdev_rouge1
```

```
HTTP 401 Unauthorized
Allow: GET, OPTIONS
Content-Type: application/json
Vary: Accept
WWW-Authenticate: Token

{
    "detail": "Authentication credentials were not provided."
}
```

15. ***Note that with the use of API token for authentication, only user with the required API token is able to connect and register the device to the website.***

16. Download the following Arduino code zipped folder ege356_lab5_wifi_http_mqtt_token.zip from Brightspace, and extract the contents to the Arduino folder (***ege356_lab5_wifi_http_mqtt_token.ino***). Make the following changes (similar to step 9) before loading the code to the M5StickC device.



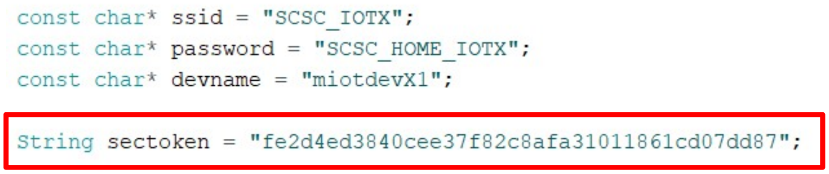
```
char MQTTServerip[] = "192.168.23.1";
String HTTPServerip = "192.168.23.1";

WiFiClient espclient;
PubSubClient client(espclient);

HTTPClient http;

const char* ssid = "SCSC_IOTX";
const char* password = "SCSC_HOME_IOTX";
const char* devname = "miotdevX1";
```

IP address should be Learner's IoT gateway IP address

SSID and password should learner's IoT gateway

Replace the X with learner's serial number on the attendance list and append a 'T' at the end, .e.g miotdev101T (if serial number is 10 on attendance list)

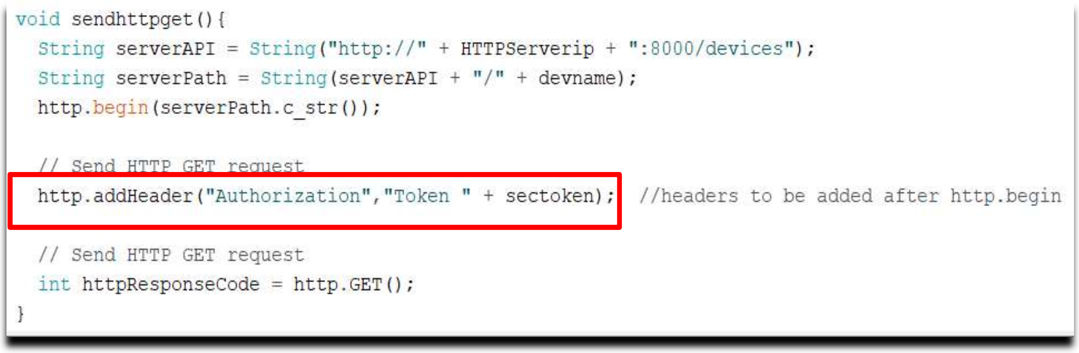
17. Note the following ***arduino codes*** added to access the WebService ***using the API token***.



```
const char* ssid = "SCSC_IOTX";
const char* password = "SCSC_HOME_IOTX";
const char* devname = "miotdevX1";

String sectoken = "fe2d4ed3840cee37f82c8afa31011861cd07dd87";
```

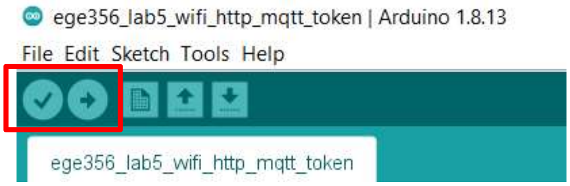
. . . . .

. . . . .

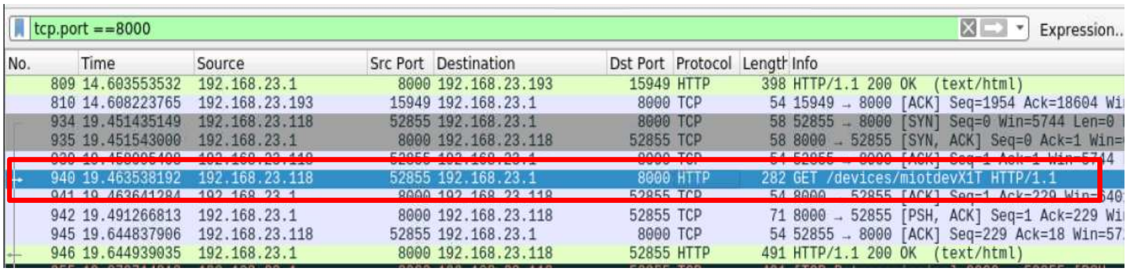


```
void sendhttpget(){
  String serverAPI = String("http://" + HTTPServerip + ":8000/devices");
  String serverPath = String(serverAPI + "/" + devname);
  http.begin(serverPath.c_str());

  // Send HTTP GET request
  http.addHeader("Authorization","Token " + sectoken);  //headers to be added after http.begin

  // Send HTTP GET request
  int httpResponseCode = http.GET();
}
```

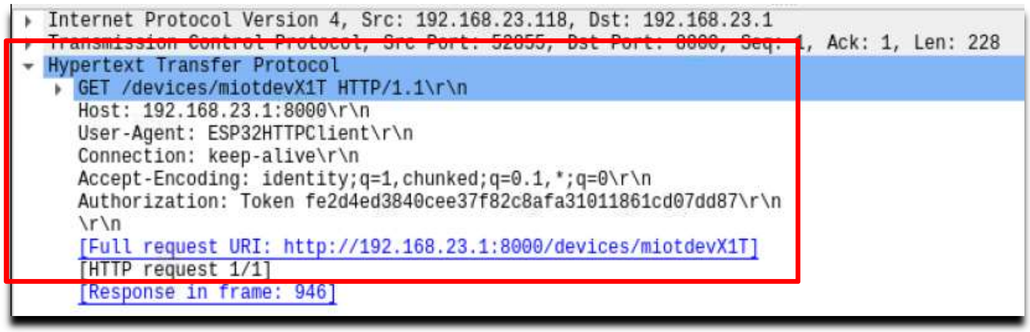
18. Compile the code, and if there are no errors, load the code to the M5StickC device.


ege356_lab5_wifi_http_mqtt_token | Arduino 1.8.13
File Edit Sketch Tools Help
ege356_lab5_wifi_http_mqtt_token

19. Start the network capture and press Button A to connect to the gateway and access the webservice.

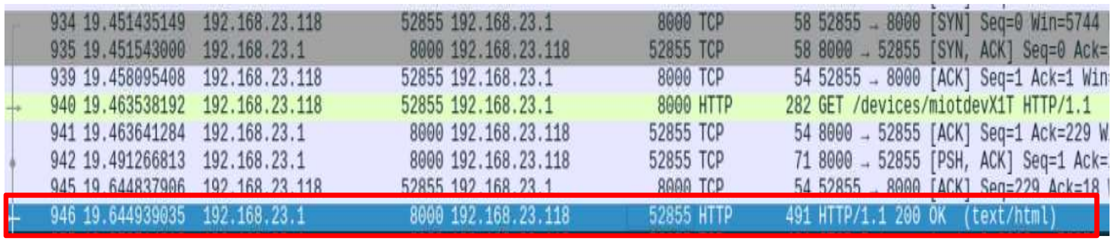


```
tcp.port ==8000
No.  Time            Source          Src Port Destination     Dst Port Protocol Length Info
 809 14.603553532  192.168.23.1       8000 192.168.23.193    15949 HTTP     398 HTTP/1.1 200 OK  (text/html)
 810 14.608223765  192.168.23.193    15949 192.168.23.1       8000 TCP       54 15949 → 8000 [ACK] Seq=1954 Ack=18604 Wi
 934 19.451435149  192.168.23.118    52855 192.168.23.1       8000 TCP       58 52855 → 8000 [SYN] Seq=0 Win=5744 Len=0
 935 19.451543000  192.168.23.1       8000 192.168.23.118    52855 TCP       58 8000 → 52855 [SYN, ACK] Seq=0 Ack=1 Win=
 939 19.458095408  192.168.23.118    52855 192.168.23.1       8000 TCP       54 52855 → 8000 [ACK] Seq=1 Ack=1 Win=5744
 940 19.463538192  192.168.23.118    52855 192.168.23.1       8000 HTTP     282 GET /devices/miotdevX1T HTTP/1.1
 941 19.463641284  192.168.23.1       8000 192.168.23.118    52855 TCP       54 8000 → 52855 [ACK] Seq=1 Ack=229 Win=640
 942 19.491266813  192.168.23.1       8000 192.168.23.118    52855 TCP       71 8000 → 52855 [PSH, ACK] Seq=1 Ack=229 Wi
 945 19.644837906  192.168.23.118    52855 192.168.23.1       8000 TCP       54 52855 → 8000 [ACK] Seq=229 Ack=18 Win=57
 946 19.644939035  192.168.23.1       8000 192.168.23.118    52855 HTTP     491 HTTP/1.1 200 OK  (text/html)
```

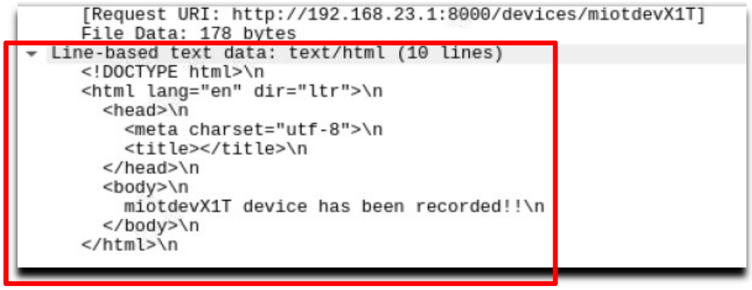


```
▸ Internet Protocol Version 4, Src: 192.168.23.118, Dst: 192.168.23.1
▸ Transmission Control Protocol, Src Port: 52855, Dst Port: 8000, Seq: 1, Ack: 1, Len: 228
▾ Hypertext Transfer Protocol
  ▸ GET /devices/miotdevX1T HTTP/1.1\r\n
    Host: 192.168.23.1:8000\r\n
    User-Agent: ESP32HTTPClient\r\n
    Connection: keep-alive\r\n
    Accept-Encoding: identity;q=1,chunked;q=0.1,*;q=0\r\n
    Authorization: Token fe2d4ed3840cee37f82c8afa31011861cd07dd87\r\n
    \r\n
    [Full request URI: http://192.168.23.1:8000/devices/miotdevX1T]
    [HTTP request 1/1]
    [Response in frame: 946]
```

20. Note the response from webservice to the M5StickC device. ***It is similar to the response in Lab4, except that it is not displayed in Serial Monitor of the M5StickC devices.***

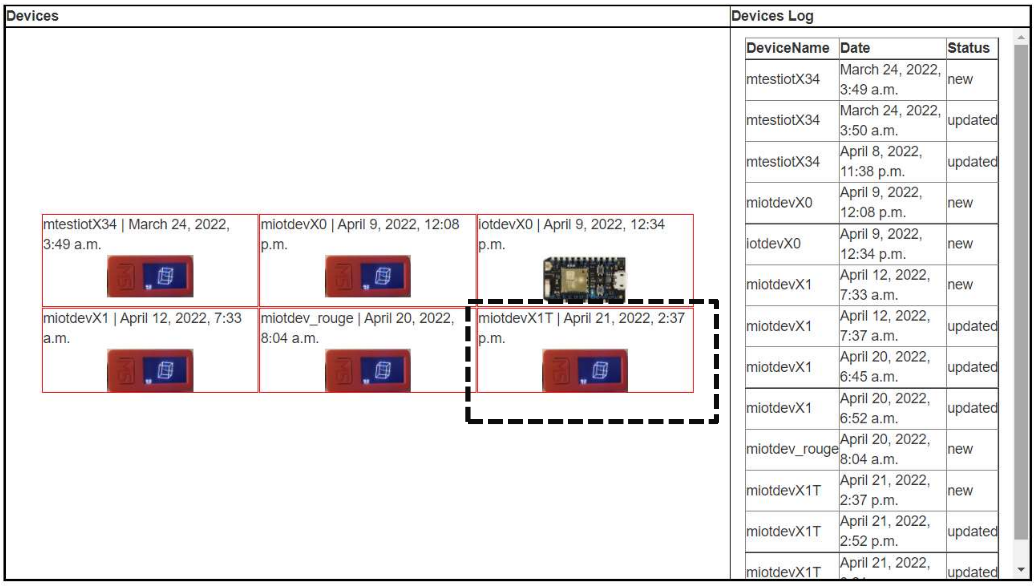


```
 934 19.451435149  192.168.23.118    52855 192.168.23.1       8000 TCP       58 52855 → 8000 [SYN] Seq=0 Win=5744
 935 19.451543000  192.168.23.1       8000 192.168.23.118    52855 TCP       58 8000 → 52855 [SYN, ACK] Seq=0 Ack=
 939 19.458095408  192.168.23.118    52855 192.168.23.1       8000 TCP       54 52855 → 8000 [ACK] Seq=1 Ack=1 Win
 940 19.463538192  192.168.23.118    52855 192.168.23.1       8000 HTTP     282 GET /devices/miotdevX1T HTTP/1.1
 941 19.463641284  192.168.23.1       8000 192.168.23.118    52855 TCP       54 8000 → 52855 [ACK] Seq=1 Ack=229 W
 942 19.491266813  192.168.23.1       8000 192.168.23.118    52855 TCP       71 8000 → 52855 [PSH, ACK] Seq=1 Ack=
 945 19.644837906  192.168.23.118    52855 192.168.23.1       8000 TCP       54 52855 → 8000 [ACK] Seq=229 Ack=18
 946 19.644939035  192.168.23.1       8000 192.168.23.118    52855 HTTP     491 HTTP/1.1 200 OK  (text/html)
```



```
    [Request URI: http://192.168.23.1:8000/devices/miotdevX1T]
    File Data: 178 bytes
▾ Line-based text data: text/html (10 lines)
    <!DOCTYPE html>\n
    <html lang="en" dir="ltr">\n
      <head>\n
        <meta charset="utf-8">\n
        <title></title>\n
      </head>\n
      <body>\n
        miotdevX1T device has been recorded!!\n
      </body>\n
    </html>\n
```

21. Observe that the web application now displays the new device.



Devices

mtestiotX34 | March 24, 2022, 3:49 a.m.

miotdevX0 | April 9, 2022, 12:08 p.m.

iotdevX0 | April 9, 2022, 12:34 p.m.

miotdevX1 | April 12, 2022, 7:33 a.m.

miotdev_rouge | April 20, 2022, 8:04 a.m.

miotdevX1T | April 21, 2022, 2:37 p.m.

Devices Log

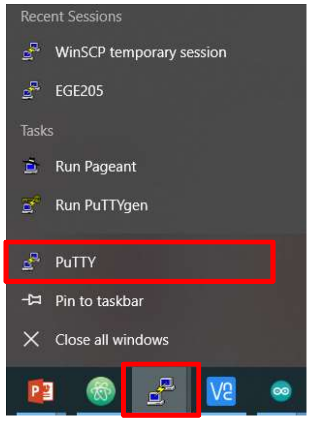
DeviceName	Date	Status
mtestiotX34	March 24, 2022, 3:49 a.m.	new
mtestiotX34	March 24, 2022, 3:50 a.m.	updated
mtestiotX34	April 8, 2022, 11:38 p.m.	updated
miotdevX0	April 9, 2022, 12:08 p.m.	new
iotdevX0	April 9, 2022, 12:34 p.m.	new
miotdevX1	April 12, 2022, 7:33 a.m.	new
miotdevX1	April 12, 2022, 7:37 a.m.	updated
miotdevX1	April 20, 2022, 6:45 a.m.	updated
miotdevX1	April 20, 2022, 6:52 a.m.	updated
miotdev_rouge	April 20, 2022, 8:04 a.m.	new
miotdevX1T	April 21, 2022, 2:37 p.m.	new
miotdevX1T	April 21, 2022, 2:52 p.m.	updated
miotdevX1T	April 21, 2022,	updated

22. If the previous Arduino code (without the token) was used, the M5StickC device would not be able to access the webservice. Likewise, keying in the URI alone to the browser to access the webservice did not work as in step 14.

23. Note that although the token authentication enabled better security, it is still not secured, as can be seen from wireshark packet analysis, that the authorization token is displayed in clear text.

Note that curl commands can be used to access the web service with the api tokens. As such, attacks can be automated

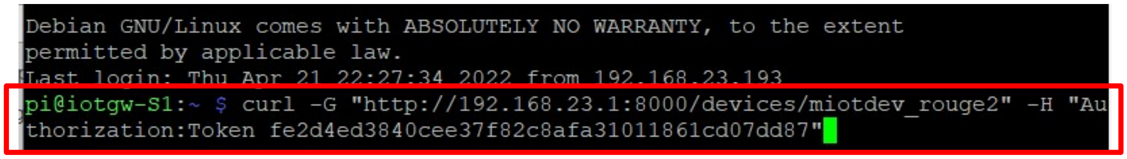
24. Base on the network packet data captured, use the following curl commands to register a new device to the web application. ***To use curl commands, connect to Rpi via a new putty session (right click on current icon and select putty to start a new session***.



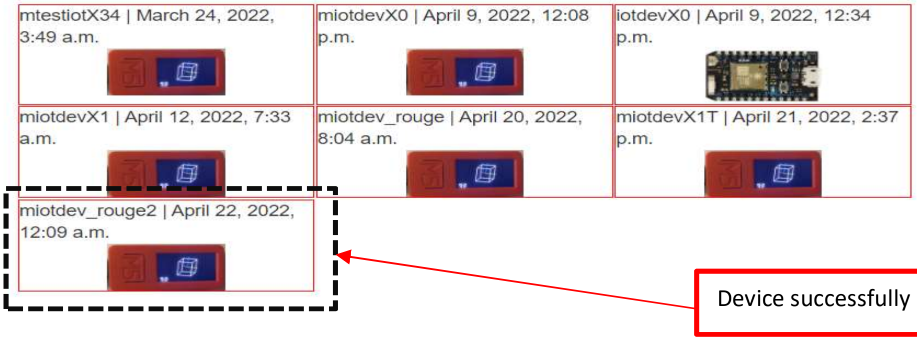
Instructor can demo here to show how the use of curl commands can automatically generate multiple device registrations to the website once the API token and access method has been compromised.

25. Key in the following command to the console as shown.

```
curl -G "http://192.168.23.1:8000/devices/miotdev_rouge2" -H "Authorization:Token fe2d4ed3840cee37f82c8afa31011861cd07dd87"
```



```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Apr 21 22:27:34 2022 from 192.168.23.193
pi@iotgw-S1:~ $ curl -G "http://192.168.23.1:8000/devices/miotdev_rouge2" -H "Au
thorization:Token fe2d4ed3840cee37f82c8afa31011861cd07dd87"
```



mtestiotX34 | March 24, 2022, 3:49 a.m.

miotdevX0 | April 9, 2022, 12:08 p.m.

iotdevX0 | April 9, 2022, 12:34 p.m.

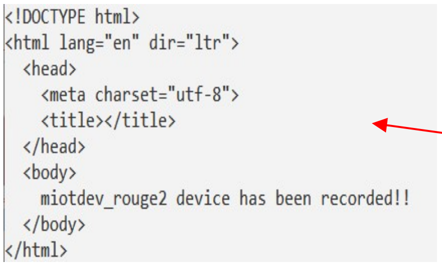
miotdevX1 | April 12, 2022, 7:33 a.m.

miotdev_rouge | April 20, 2022, 8:04 a.m.

miotdevX1T | April 21, 2022, 2:37 p.m.

miotdev_rouge2 | April 22, 2022, 12:09 a.m.

Device successfully registered.



```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    miotdev_rouge2 device has been recorded!!
  </body>
</html>
```

Response from web service on successful device registration.

26. Using the following curl commands, access the https APIs from Adafruit IO and do a network capture.

```
curl -G "https://io.adafruit.com/api/v2/username/feeds/lab1-led" -H "X-AIO-Key: aio_fBdsdxiqhwXH7NbU1D0exuQ3442f"
```

Follow the steps:

**** 1. type in the curl command to the putty console session as previously done**

**** 2. replace the username with learner's Adafruit IO username**

**** 3. replace the X-AIO-Key value with learner's Adafruit API Token value.**

**** 4. Note space between X-AIO-Key: and the Token value**

X-AIO-Key: aio_fBdsdxiqhwXH7NbU1D0exuQ3442f"

If unsure about step 6, get help from the instructor.

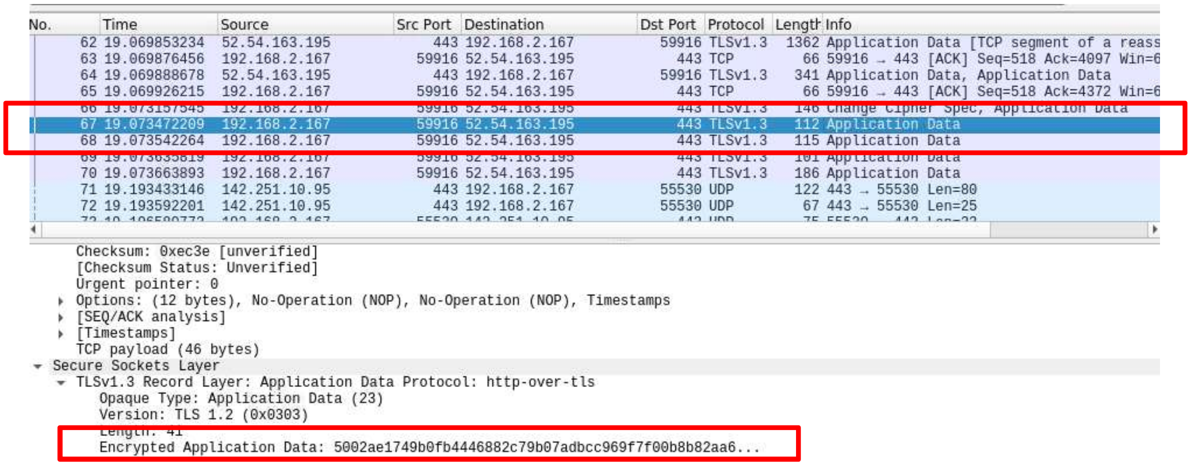
**** 5. Clear the contents in the display filter of wireshark if any**

**** 6. Select the wlan0 interface (as data flow is through the wlan0 interface)**

**** 7. start the network capture before executing the command**

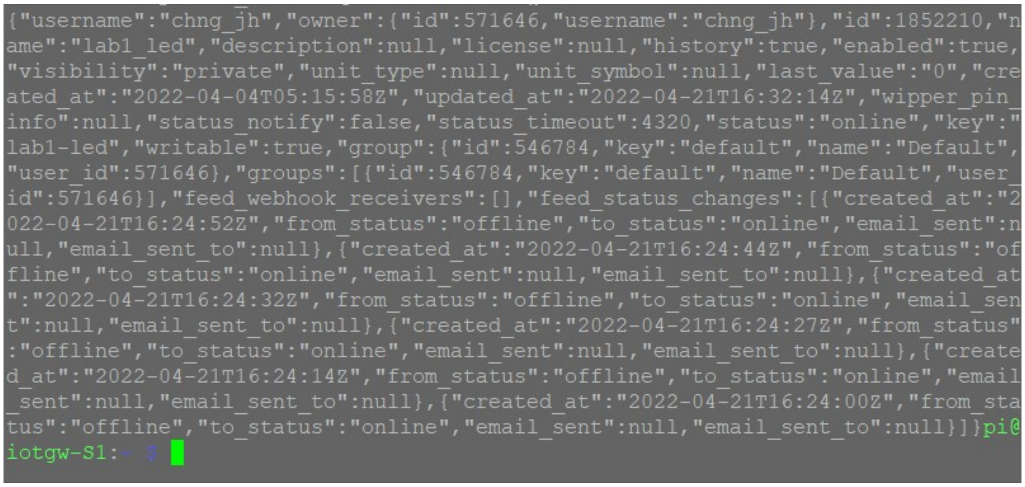
**** 8. Execute the curl command.**

27. Learners should observe the following results that the data is encrypted and the API token cannot be simply retrieved by reading out the data.



```
No.   Time          Source          Src Port Destination     Dst Port Protocol Length Info
  62 19.069853234  52.54.163.195       443 192.168.2.167     59916 TLSv1.3  1362 Application Data [TCP segment of a reass
  63 19.069876456  192.168.2.167     59916 52.54.163.195       443 TCP        66 59916 → 443 [ACK] Seq=518 Ack=4097 Win=6
  64 19.069888678  52.54.163.195       443 192.168.2.167     59916 TLSv1.3   341 Application Data, Application Data
  65 19.069926215  192.168.2.167     59916 52.54.163.195       443 TCP        66 59916 → 443 [ACK] Seq=518 Ack=4372 Win=6
  66 19.073157545  192.168.2.167     59916 52.54.163.195       443 TLSv1.3   146 Change Cipher Spec, Application Data
  67 19.073472209  192.168.2.167     59916 52.54.163.195       443 TLSv1.3   112 Application Data
  68 19.073542264  192.168.2.167     59916 52.54.163.195       443 TLSv1.3   115 Application Data
  69 19.073635819  192.168.2.167     59916 52.54.163.195       443 TLSv1.3   101 Application Data
  70 19.073663893  192.168.2.167     59916 52.54.163.195       443 TLSv1.3   186 Application Data
  71 19.193433146  142.251.10.95       443 192.168.2.167     55530 UDP       122 443 → 55530 Len=80
  72 19.193592201  142.251.10.95       443 192.168.2.167     55530 UDP        67 443 → 55530 Len=25
  73 19.196580773  192.168.2.167     55530 142.251.10.95       443 UDP        75 55530 → 443 Len=33

    Checksum: 0xec3e [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
    TCP payload (46 bytes)
▾ Secure Sockets Layer
  ▾ TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
      Opaque Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 41
      Encrypted Application Data: 5002ae1749b0fb4446882c79b07adbcc969f7f00b8b82aa6...
```

Console display would be as shown:



```
{"username":"chng_jh","owner":{"id":571646,"username":"chng_jh"},"id":1852210,"n
ame":"lab1_led","description":null,"license":null,"history":true,"enabled":true,
"visibility":"private","unit_type":null,"unit_symbol":null,"last_value":"0","cre
ated_at":"2022-04-04T05:15:58Z","updated_at":"2022-04-21T16:32:14Z","wipper_pin_
info":null,"status_notify":false,"status_timeout":4320,"status":"online","key":"
lab1-led","writable":true,"group":{"id":546784,"key":"default","name":"Default",
"user_id":571646},"groups":[{"id":546784,"key":"default","name":"Default","user_
id":571646}],"feed_webhook_receivers":[],"feed_status_changes":[{"created_at":"2
022-04-21T16:24:52Z","from_status":"offline","to_status":"online","email_sent":n
ull,"email_sent_to":null},{"created_at":"2022-04-21T16:24:44Z","from_status":"of
fline","to_status":"online","email_sent":null,"email_sent_to":null},{"created_at
":"2022-04-21T16:24:32Z","from_status":"offline","to_status":"online","email_sen
t":null,"email_sent_to":null},{"created_at":"2022-04-21T16:24:27Z","from_status"
:"offline","to_status":"online","email_sent":null,"email_sent_to":null},{"create
d_at":"2022-04-21T16:24:14Z","from_status":"offline","to_status":"online","email
_sent":null,"email_sent_to":null},{"created_at":"2022-04-21T16:24:00Z","from_sta
tus":"offline","to_status":"online","email_sent":null,"email_sent_to":null}]}pi@
iotgw-S1:~ $ 
```

Data is in json format.

28. Summary: In the development of web services, APIs are created for devices to access and exchange data. These APIs are provided in the form of an URI. To prevent unauthorized usage of the URIs, users must be authenticated. However, http URIs communicate in clear text and this allows attackers to capture password and authentication data(tokens) in clear text. To further secure the use of these APIs, the data should be encrypted to prevent exchange of data in clear text. It is also important to note that encryption requires additional resources and increases the latency and thus affects response time. As a result, till today, there are still many OT and IoT devices exchanging data over the network without encryption.