COMSM0119_2024_TB2

Internet of Things & Industrial IOT

Group Assignment

Topic:
Vulnerabilities of IOT's

Student No: 2451975

Faculty of Engineering
School of Computer Science
University of Bristol



Contents

1 Introduction **3**

2 Host Discovery **4**

3 Attacks deployed **7**

4 APK vulnerability **13**

5 Mitigations **16**

5.1 Host Discovery Mitigations 16

5.2 Attacks Deployed (Mitigations) 16

5.3 Strategies to Mitigate MQTT Man-in-the-Middle (MiTM) Attacks 17

6 Conclusion **17**

7 References **17**

1 Introduction

The Internet-of-Things (IoT) is an emerging paradigm due to extensive developments in information and communication technology (ICT). The purpose of IoT is to expand the functions of the first version of the Internet by increasing the ability to connect numerous objects. The IoT model has expanded to span different applications such as manufacturing and Industry 4.0, eHealth, smart cities and homes, robotics and drones, transportation, and critical infrastructures[1].

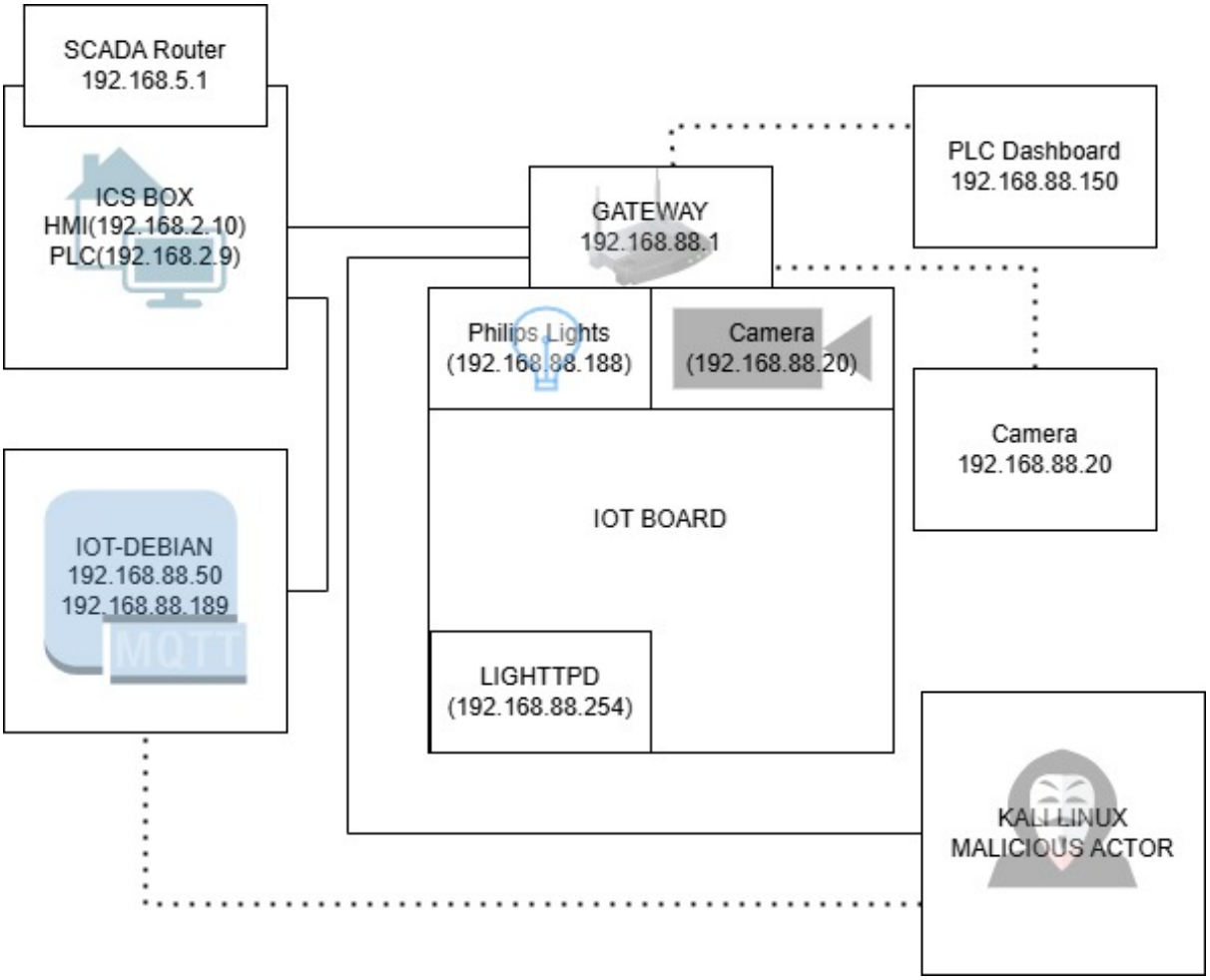


Figure 1: Network Architecture Diagram

2 Host Discovery

```
Nmap scan report for 192.168.88.1
Host is up (0.0032s latency).

PORT     STATE    SERVICE        VERSION
22/tcp   open     ssh            MikroTik RouterOS sshd (protocol 2.0)
23/tcp   open     telnet         Linux telnetd
53/tcp   filtered domain
80/tcp   filtered http
161/tcp  filtered snmp
443/tcp  filtered https
502/tcp  filtered mbap
1883/tcp filtered mqtt
8080/tcp filtered http-proxy
8883/tcp filtered secure-mqtt
8888/tcp filtered sun-answerbook
Service Info: OS: Linux; Device: router; CPE: cpe:/o:mikrotik:routeros, cpe:/
o:linux:linux_kernel
```

Figure 2: Gateway

```
Nmap scan report for 192.168.5.1
Host is up (0.23s latency).
Not shown: 992 closed tcp ports (reset)
PORT     STATE    SERVICE        VERSION
21/tcp   open     ftp            MikroTik router ftpd 6.49.17
22/tcp   open     ssh            MikroTik RouterOS sshd (protocol 2.0)
23/tcp   open     telnet         Linux telnetd
53/tcp   open     domain         (generic dns response: NOTIMP)
80/tcp   open     http           MikroTik router config httpd
514/tcp  filtered shell
2000/tcp open     bandwidth-test MikroTik bandwidth-test server
8291/tcp open     unknown
1 service unrecognized despite returning data. If you know the service/version, please submit the following finge
rprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port53-TCP:V=7.94SVN%I=7%D=3/27%Time=67E534C6%P=x86_64-pc-linux-gnu%r(D
SF:NSVersionBindReqTCP,E,"\0\x0c\0\x06\x81\x84\0\0\0\0\0\0\0");
Aggressive OS guesses: Actiontec MI424WR-GEN3I WAP (99%), DD-WRT v24-sp2 (Linux 2.4.37) (98%), Linux 3.2 (96%), M
icrosoft Windows XP SP3 or Windows 7 or Windows Server 2012 (96%), Linux 4.4 (94%), Microsoft Windows XP SP3 (94%
), BlueArc Titan 2100 NAS device (90%)
No exact OS matches for host (test conditions non-ideal).
Service Info: Host: MikroTik MSc; OSs: Linux, RouterOS; Device: router; CPE: cpe:/o:mikrotik:routeros, cpe:/o:lin
ux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 282.63 seconds
```

Figure 3: Scada Router Ip

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-28 07:56 EDT
Nmap scan report for 192.168.88.20
Host is up (0.00091s latency).
Not shown: 996 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT     STATE SERVICE
22/tcp   open  ssh
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
8081/tcp open  blackice-icecap
```

Figure 4: Camera Ip

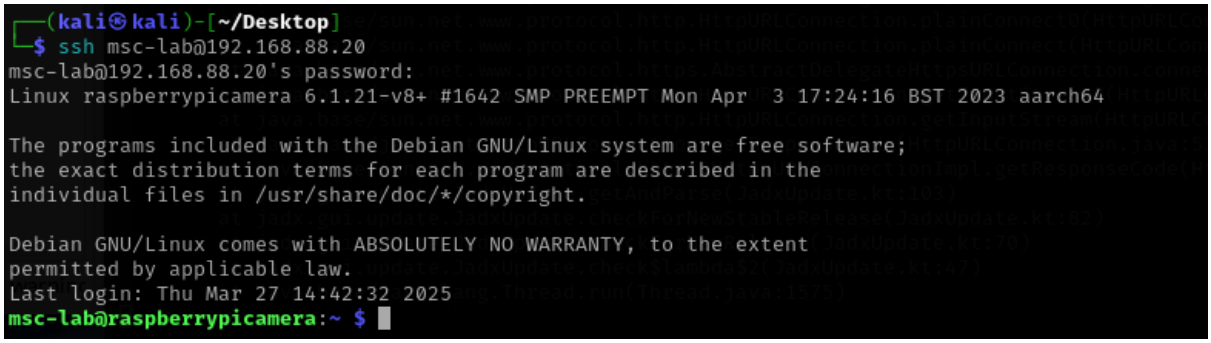


Figure 5: SSH login in Camera

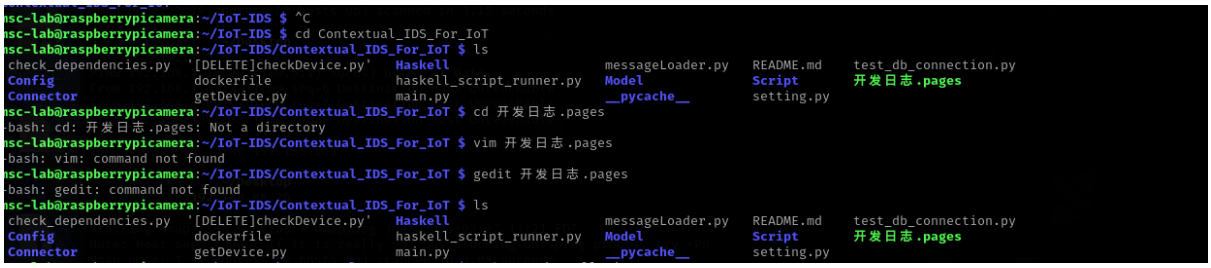


Figure 6: Camera Raspberry files

As I had gateway ip seen in Figure 2 I logged in into router we could see the ip 192.168.88.50 which had a open port ssh. we tried to login with msc-lab and the Password as MScCyb3rS3c. and we got in.

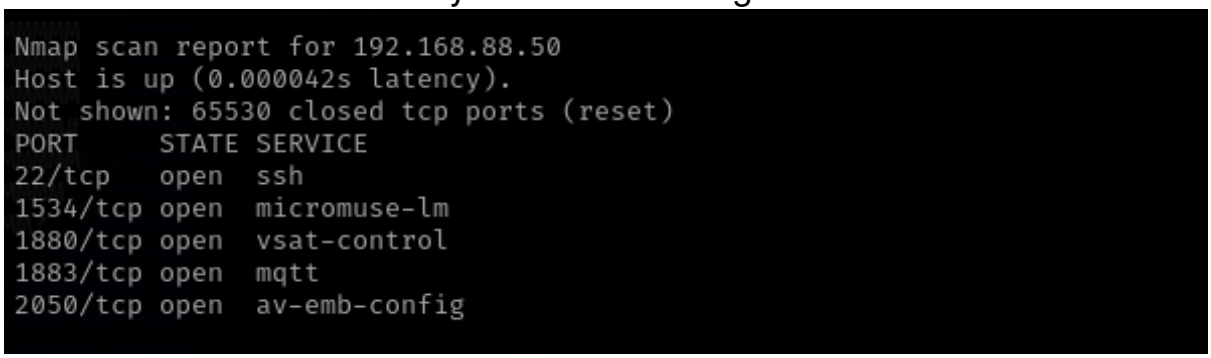


Figure 7: Open port scan for 192.268.88.50

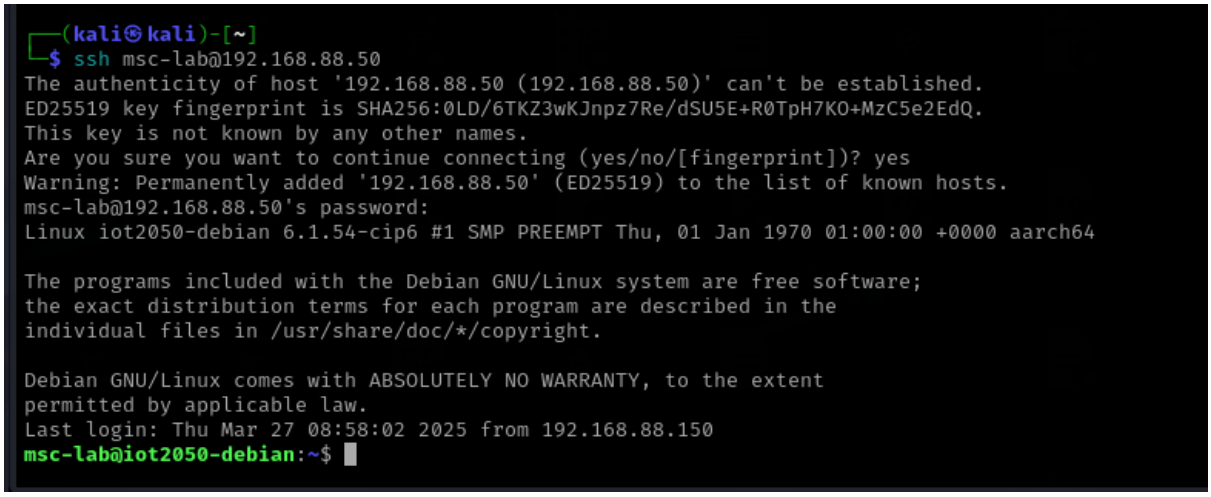


Figure 8: SSH login successful

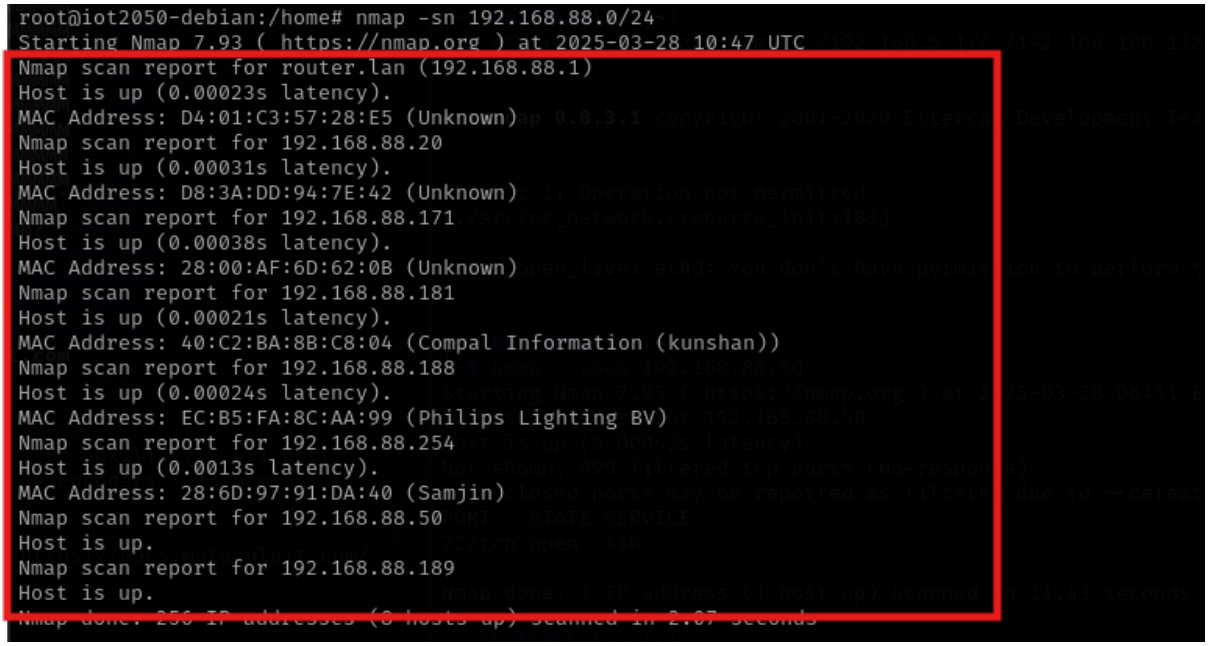


Figure 9: Nmap to find the connected IOT's

```
Nmap scan report for 192.168.88.171
Host is up (0.00047s latency).
Not shown: 65511 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
808/tcp   open  ccproxy-http
1801/tcp  open  msmq
2103/tcp  open  zephyr-clt
2105/tcp  open  eklogin
2107/tcp  open  msmq-mgmt
2179/tcp  open  vmrdp
4002/tcp  open  mlchat-proxy
4410/tcp  open  itwo-server
7680/tcp  open  pando-pub
11885/tcp open  unknown
11886/tcp open  unknown
11887/tcp open  unknown
11888/tcp open  unknown
22350/tcp open  CodeMeter
22352/tcp open  unknown
27000/tcp open  flexlm0
49671/tcp open  unknown
49684/tcp open  unknown
49685/tcp open  unknown
49722/tcp open  unknown
56658/tcp open  unknown
MAC Address: 28:00:AF:6D:62:0B (Unknown)
```

Figure 10: Open ports for 192.168.88.171

```
Nmap scan report for 192.168.88.181
Host is up (0.00052s latency).
Not shown: 65534 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT     STATE SERVICE
7680/tcp open  pando-pub
MAC Address: 40:C2:BA:8B:C8:04 (Compal Information (kunshan))
```

Figure 11: Open ports scan for 192.168.88.181

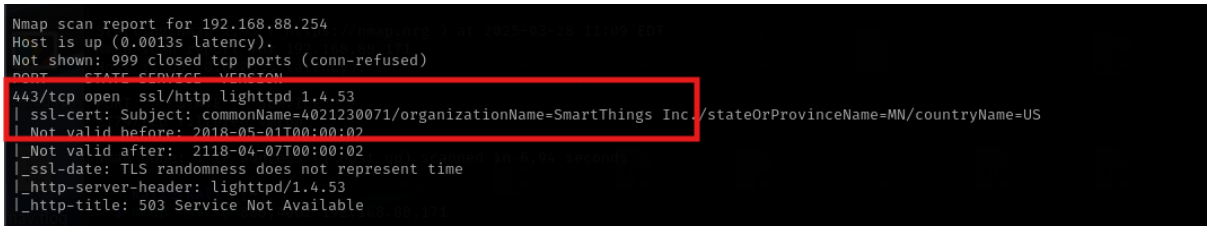


Figure 12: Smart things IP

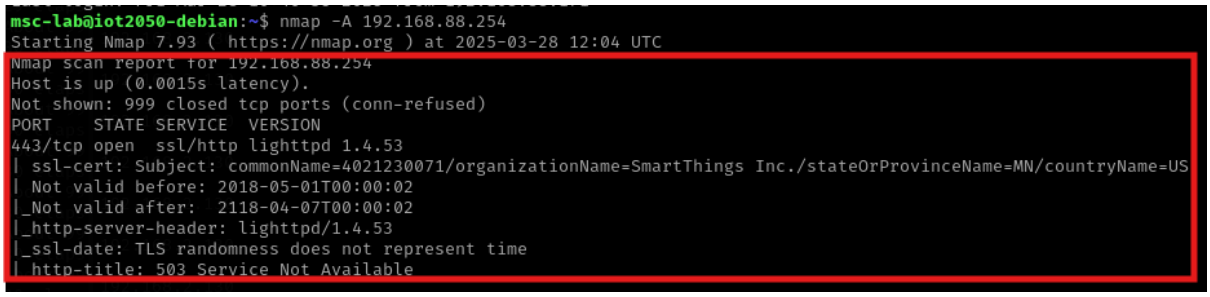


Figure 13: Open ports for 192.168.88.254

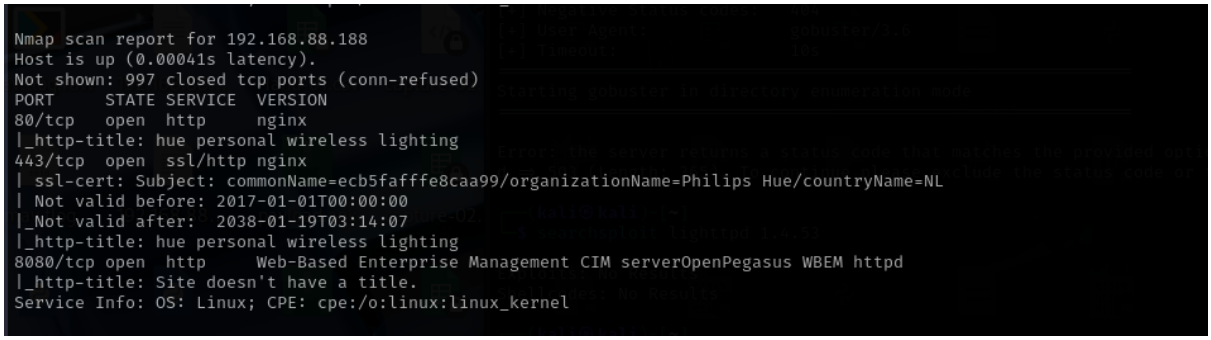


Figure 14: Open ports for 192.168.88.188

Open Ports: 80/tcp (nginx)
SSL Certificate Details:
Common Name (CN): ecb5fafffe8caa99
Organization: Philips Hue
Country: Netherlands (NL)
Valid from: 2017 $\rightarrow$ Expires in 2038
It also has open 8080 port

3 Attacks deployed

After Scanning for open ports for the Ip 192.168.88.50. I got to know that it has mqtt (1883) port open. By using this I remember that we were taught how to do MiTM attack using MQTT.
I quickly typed this command **mosquitto_pub -h 192.168.88.50 -t 'test/-topic' -m 'malicious_message'** as shown in the figure 15 so that I can see that I can pass malicious messages/ do MiTM attack. And I was successful

as shown in figure 16 where we can see that traffic for MQTT port and the message we sent. If we do follow to the stream we can actually see the full message that can be seen in the figure 17. After performing a deep nmap I found that 192.168.88.189 and 192.168.88.50 these two IPs has the same ssh Keys that they share.



Figure 15: Mosquitto MiTM attack

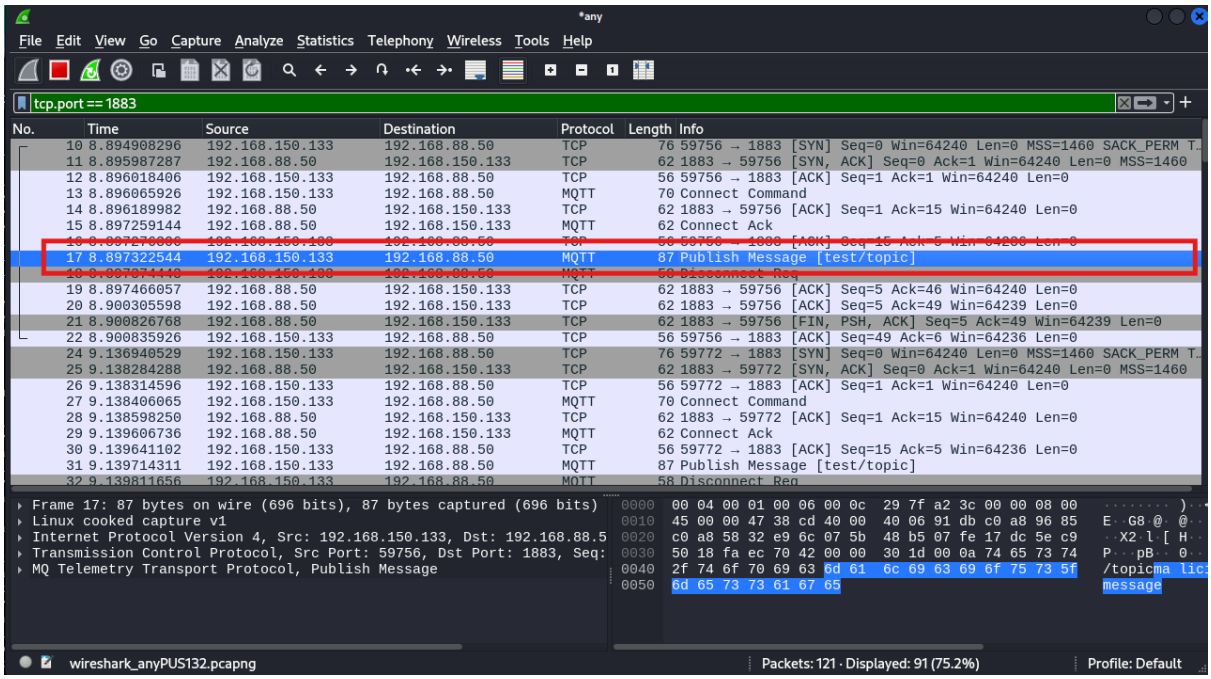


Figure 16: Wireshark traffic for MQTT

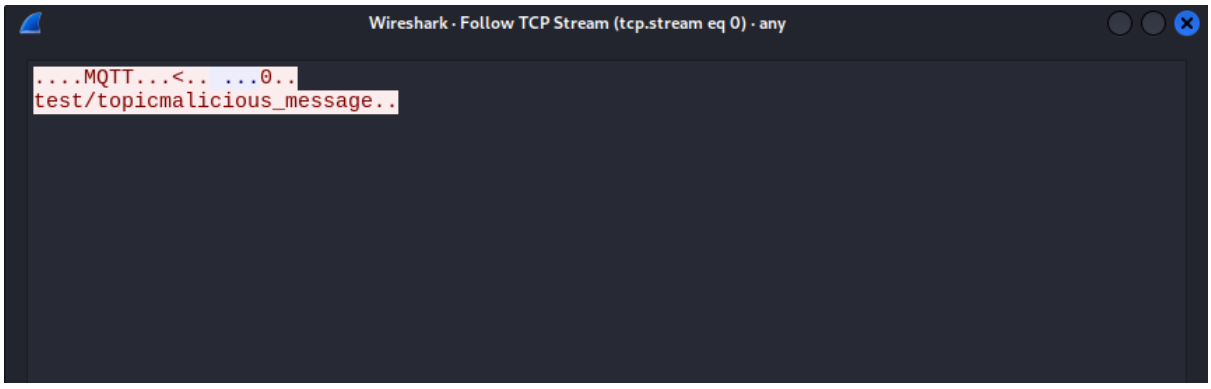


Figure 17: Full malicious message

I also got an opportunity to do the Api attack where I found the ip of the Philips Hue lights i.e 192.168.88.188 where we can see there is open port 80 as shown in the figure 18. I quickly typed the ip in the browser and I got a simple page of Philips site. I checked for any api or remote control

using the light ip. I got a official link for the Philips hue where we can debug their products [2]. This official site tells us to use debug webpage that is `https://<bridge ip address>/debug/clip.html` By going to this webpage I found that I needed to get the api of the device to get the api I have to put /api in the URL and press the Philips bridge button and after that we will get the api for the Light as seen in the Figure 19, after getting the api I just needed to check what is the light number in this case it was number 8 as seen in Figure 20. Then to control the light I just have to put /api/API-key/lights/8/state. In the body text box I just have to put ”bri”: 254, ”on”: true for turning on the light as seen in the Figure 22 and ”bri”: 254, ”on”: false to get the lights off. for all this commands to work I have to press PUT action button to work our remote controlling of the lights. We can see the live demonstration in Figure 23 and 24 respectively.

```
Nmap scan report for 192.168.88.188
Host is up (0.0051s latency).

PORT     STATE    SERVICE        VERSION
22/tcp   filtered ssh
23/tcp   filtered telnet
53/tcp   filtered domain
80/tcp   open     http           nginx
161/tcp  filtered snmp
443/tcp  filtered https
502/tcp  filtered mbap
1883/tcp filtered mqtt
8080/tcp open     http           Web-Based Enterprise Management CIM serverOp
enPegasus WBEM httpd
8883/tcp filtered secure-mqtt
8888/tcp filtered sun-answerbook
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 18: Philips hue ip open port

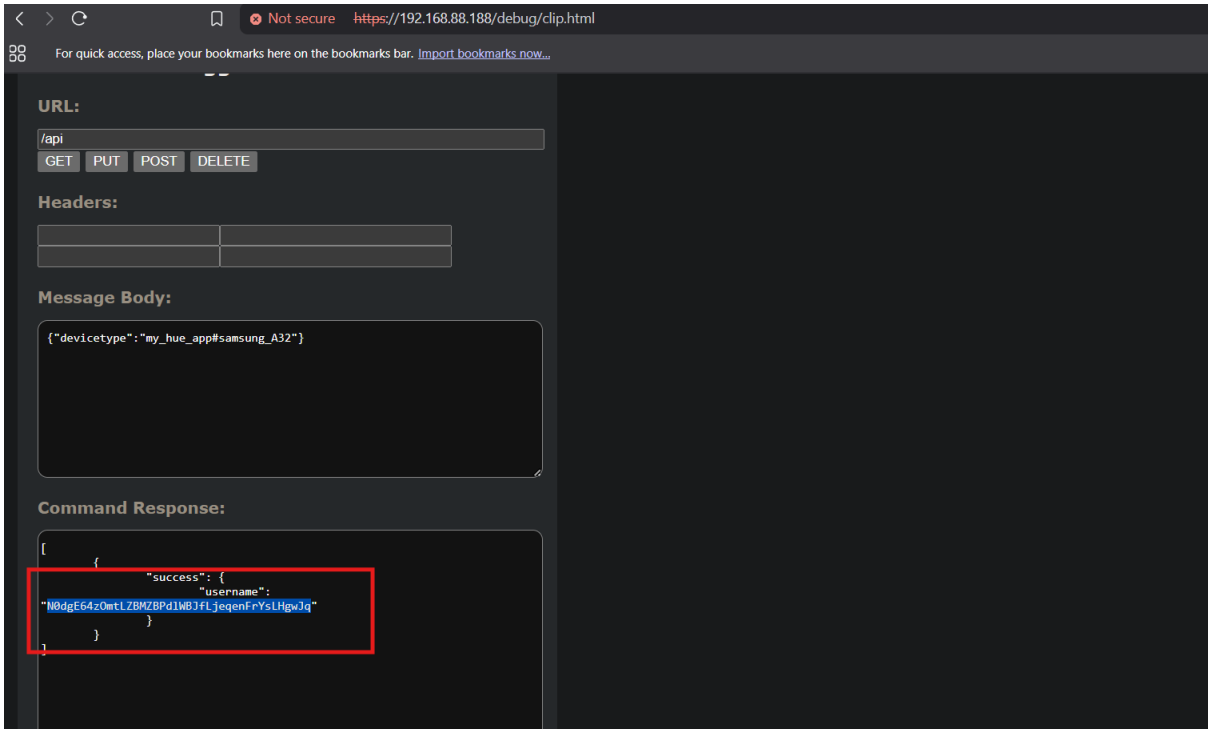


Figure 19: Api hashcode for philips

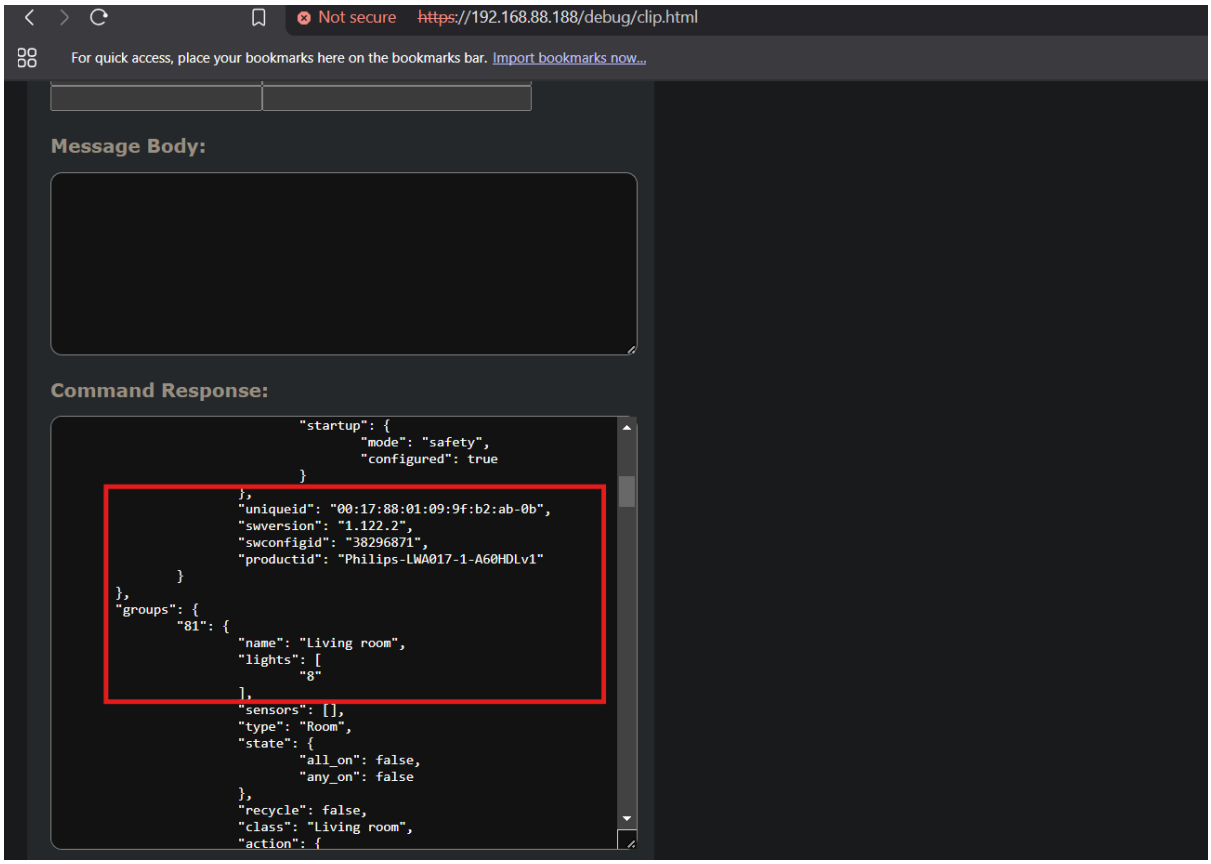


Figure 20: To know what is the lights no i.e 8

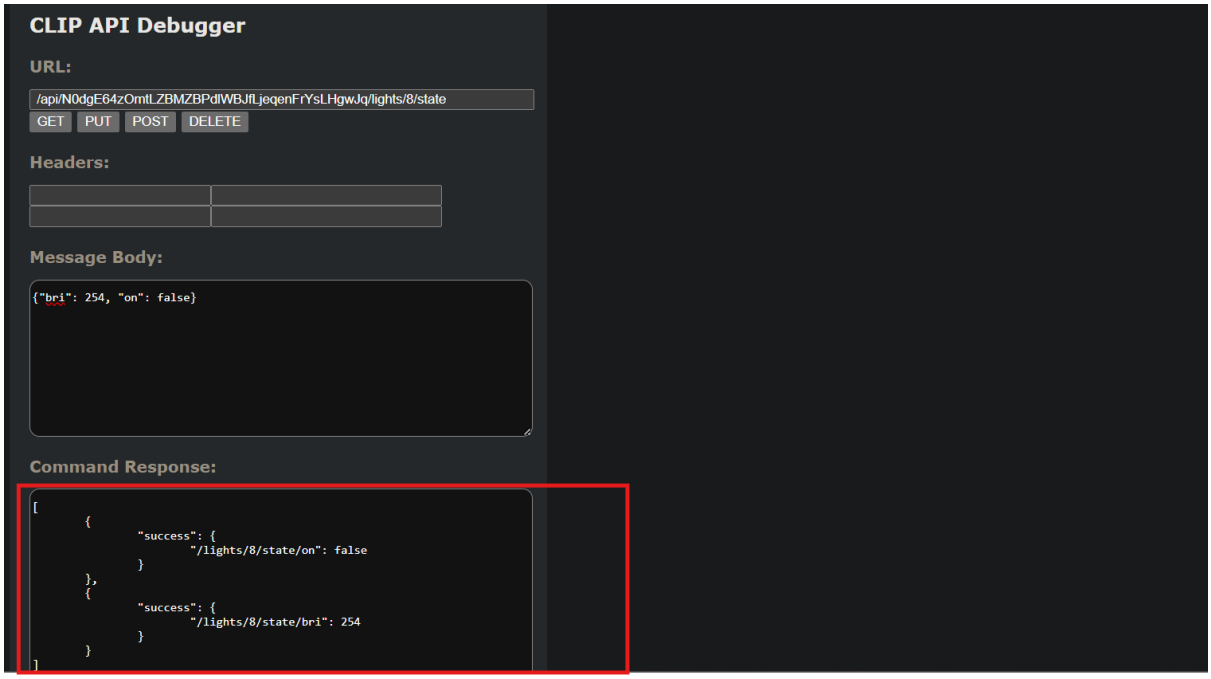


Figure 21: Philips light turned off

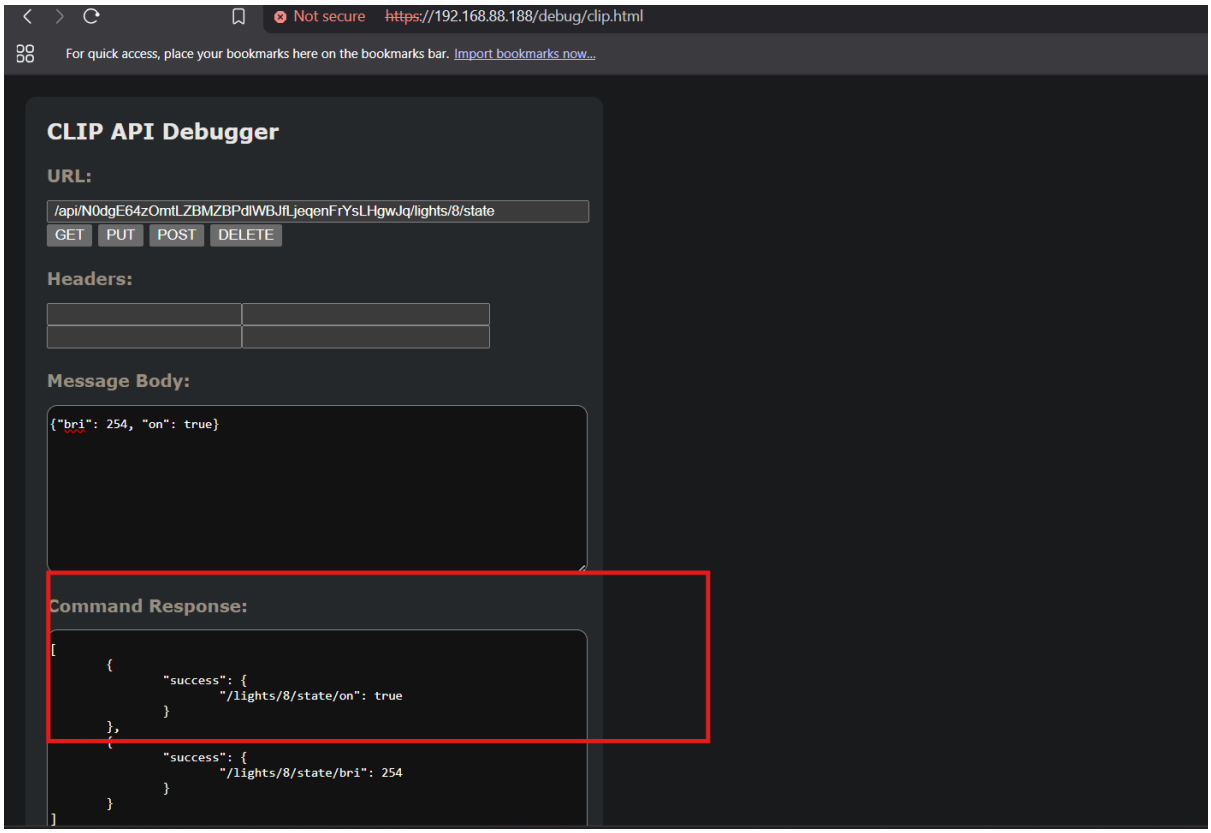


Figure 22: Philips light turned on

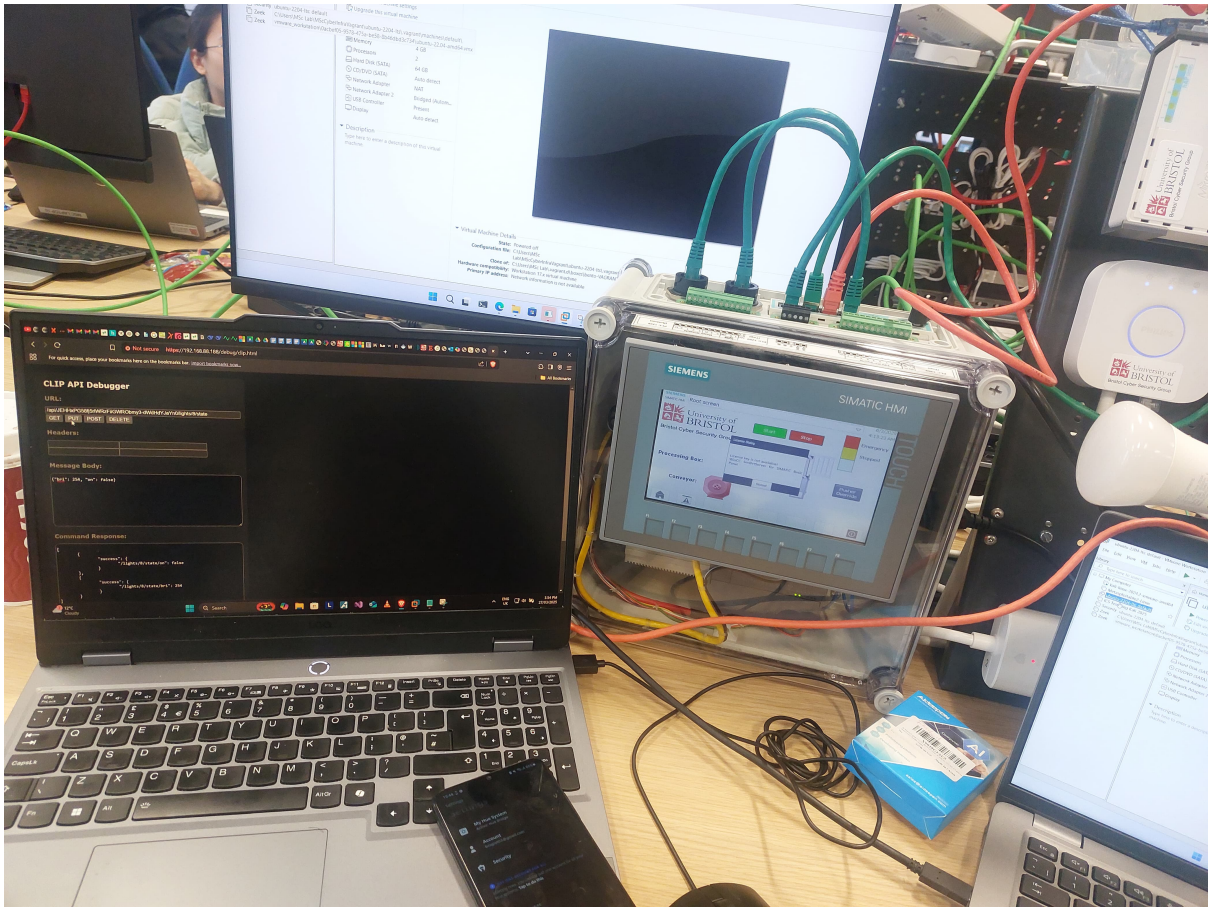


Figure 23: Lights turned off

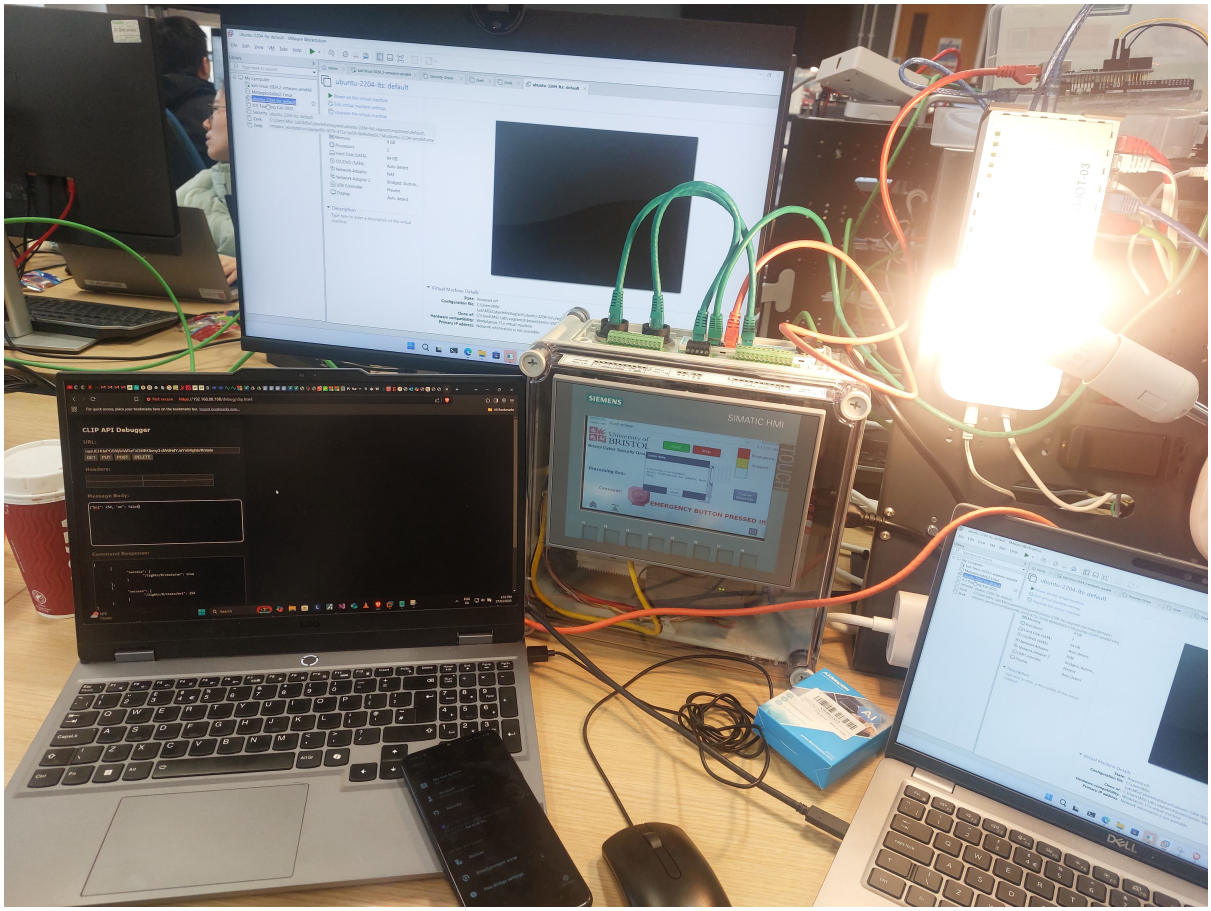


Figure 24: lights turned on

4 APK vulnerability

For APK Vulnerability I used JADX which is a powerful open-source tool for decompiling and analyzing Android apps.

```
┌──(kali㉿kali)-[~]
└─$ jadx-gui /home/kali/Desktop/App.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
INFO  - Checking for updates... Update channel: STABLE, current version: 1.5.1
INFO  - output directory: App
INFO  - loading ...
WARN  - Found duplicated class: kotlin.coroutines.jvm.internal.DebugProbesKt, count: 2. Only one will be loaded!
  classes.dex
  App.apk:DebugProbesKt.bin
INFO  - Loaded classes: 6815, methods: 50195, instructions: 1331989
INFO  - Resetting disk code cache, base dir: /home/kali/.cache/jadx/projects/App-280da7b063e9572a4101eed777ce9748/code
```

Figure 25: Jadx

android:debuggable=”true”
Risk: Debugging mode is enabled, which can expose vulnerabilities in a production environment.
Fix: Remove or set android:debuggable=”false” before release. As seen in the Figure 26.

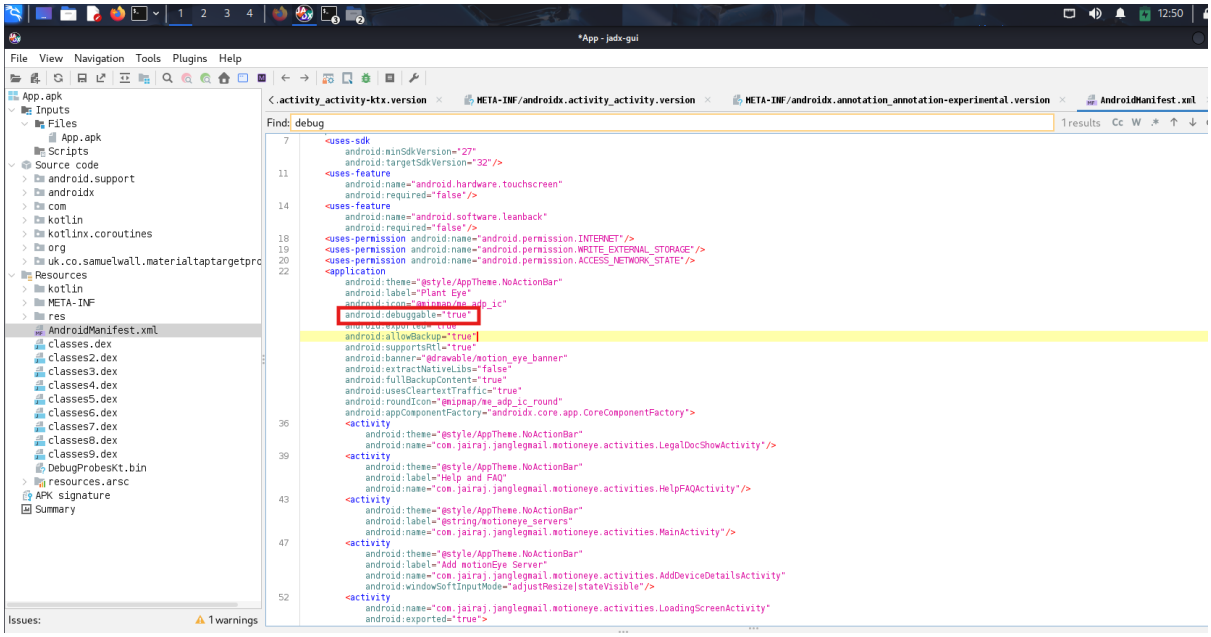


Figure 26: Android debuggable

android:usesCleartextTraffic=”true”

Risk: Allows HTTP (non-secure) connections, which could expose data to interception.

Fix: If HTTPS is available, set it to false or use a Network Security Configuration. As seen in the Figure 27.

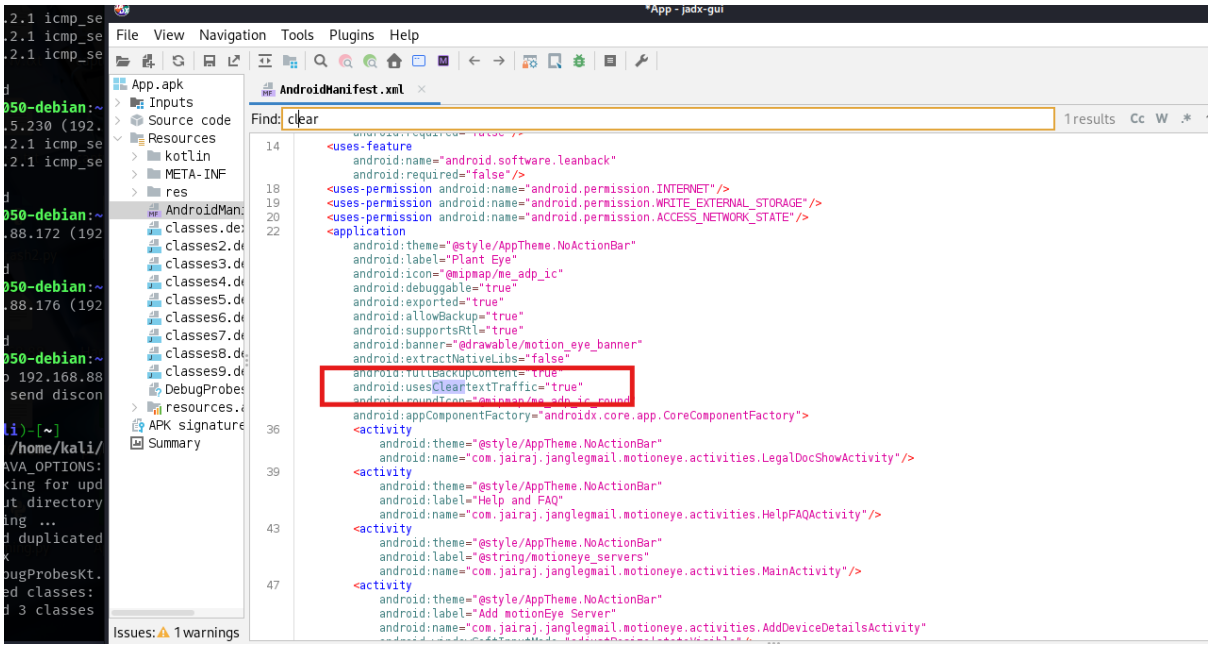


Figure 27: Allows HTTP connection

android:exported=”true”

Risk: Activities like LoadingScreenActivity and LoadingScreenTvActivity are exported, meaning other apps can potentially launch them.

Fix: Ensure that only necessary activities are exported. If not required, set

android:exported=”false”. As seen in the Figure 28.

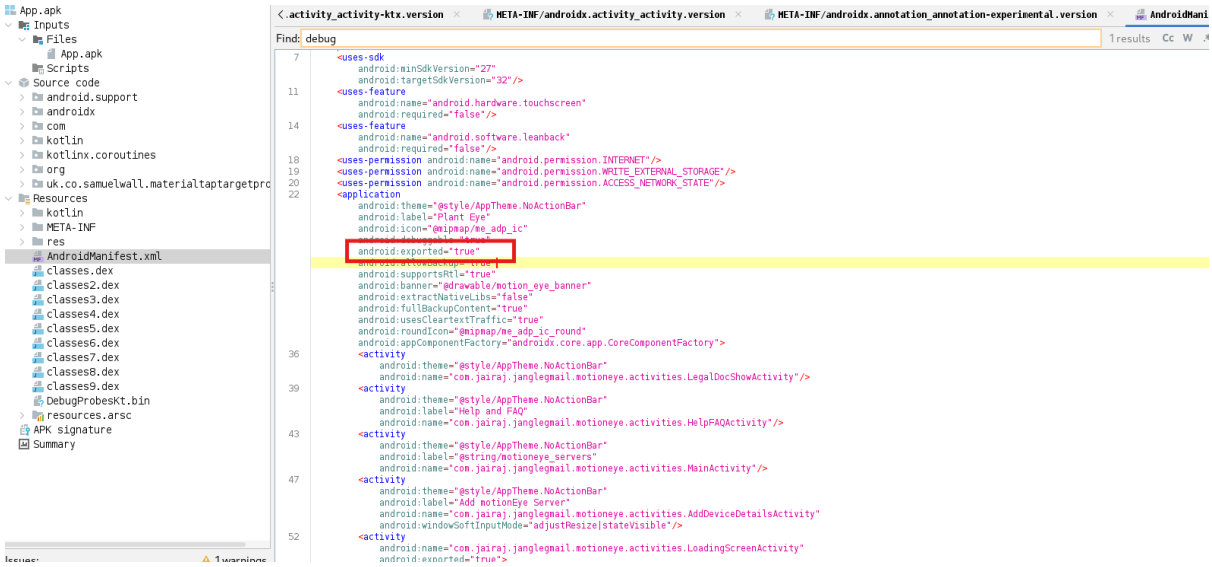


Figure 28: privacy issues

Permission:android.permission.WRITE_EXTERNAL_STORAGE
Risk: The WRITE_EXTERNAL_STORAGE permission is considered sensitive because it allows the app to write files to external storage, potentially exposing data. As seen in the Figure 29



Figure 29: APK Signature

As I checked the APK file Signature details. I found out that this is not a official release as the signature clearly states that this version is for debug. Which means this version can be tampered and can be made vulnerable to users.

5 Mitigations

5.1 Host Discovery Mitigations

Security Practice	Description
Network Segmentation	Isolate IoT devices from critical systems using VLANs or dedicated networks for enhanced security.
Firewall Rules	Limit access to sensitive services (e.g., SSH, MQTT) by restricting them to specific authorized IP ranges.
Disable Unused Services	Turn off unnecessary services like SSH or MQTT if they aren't required for the IoT device's functionality.
Use Strong Authentication	Replace default passwords with secure ones and opt for key-based authentication for accessing SSH.

Table 1: Host Discovery Mitigations

5.2 Attacks Deployed (Mitigations)

Mitigation Strategy	Description
Enable MQTT Authentication	Use username/password authentication or TLS encryption to secure communication.
Restrict Topics	Implement access control lists (ACLs) to prevent unauthorized publishing or subscribing to topics.
Monitor Traffic	Deploy intrusion detection systems (IDS) to identify unusual MQTT traffic patterns and mitigate threats.

Table 2: IoT Security Practices

5.3 Strategies to Mitigate MQTT Man-in-the-Middle (MiTM) Attacks

Mitigation Strategy	Description
Disable Debug Mode in IoT Devices	Disable public debugging interfaces such as /debug/clip.html to prevent unauthorized access.
Use Authentication Tokens	Require secure authentication tokens to access and modify API endpoints, enhancing security.
Limit Network Exposure	Block access to unnecessary API ports from the internet to reduce potential attack surfaces.

Table 3: Strategies to Mitigate Philips Hue API Attacks

6 Conclusion

The mitigation strategies for securing IoT ecosystems focus on network segmentation, strict authentication, and minimizing exposed services to reduce attack surfaces. Implementing firewalls, restricting network access, and disabling unnecessary features prevent unauthorized entry, while enforcing strong authentication, MQTT topic restrictions, and encryption secures data transmission. Addressing API vulnerabilities through authentication tokens and restricting public access further enhances security. Additionally, securing APK configurations by disabling debugging, enforcing HTTPS, and restricting exported activities prevents application-level exploits. Continuous monitoring, regular security audits, and firmware updates remain essential for maintaining robust IoT security against evolving threats.

7 References

[1] Internet of Things Security, Authors Chuan-Kun Wu (https://link-springer-com.bris.idm.oclc.org/book/10.1007/978-981-16-1372-2)

[2] Internet of Things Security and Privacy in Cyberspace, Authors Sandeep Saxena, Ashok Kumar Pradhan (https://link-springer-com.bris.idm.oclc.org/book/10.1007/978-981-19-1585-7)

[3] Security Framework and Defense Mechanisms for IoT Reactive Jamming Attacks, Authors Fadele Ayotunde Alaba , Alvaro Rocha (https://link-springer-com.bris.idm.oclc.org/book/10.1007/978-3-031-65929-4)

[3] Security Designs for the Cloud, IoT, and Social Networking, Authors Fadele Ayotunde Alaba , Alvaro Rocha (https://ebookcentral.proquest.com/lib/bristol/reader.action?docID=5940093ppg=2)