JUNE 27, 2021

IOT DEVICE SECURITY ASSESSEMNT

AN ASSESSMENT OF A VULNERABLE IOT DEVICE AS PROVIDED BY CENSIS IOT

ISAAC BASQUE-RICE
ABERTAY UNIVERSITY
Bell Street, Dundee, DD1 1HG

CONTENTS

INTRODUCTION

Access has been given by CENSIS, a non-profit sensor and IoT security firm, to a pair of boxes containing electronics powered by a USB power supply and connected to one another by an Ethernet cable running between the two boxes. The intent of this paper is to document the process of implementing a security assessment methodology on the devices.

Contents

According to information provided by CENSIS, within one of the boxes is a **NUCLEO-F746ZG** circuit board, connected to which are the ethernet and power cables. Also attached to this board is a separate board with a module called **LRWAN_GS_HF1** (a LoRaWAN Gateway device from STMicroelectronics) and an antenna sticking out of it with the label **868-915MHz**.

In the other box is a **Raspberry Pi** device connected to some other PCBs and the power and ethernet cables mentioned previously. Two of the other boards connected to the Pi are marked **Pycom LoPy4** and **Pycom Pysense**, they are also connected to one another. There are also a few additional miscellaneous boards which are not marked as well, one of them, however, is labelled **I-NUCLEO-LRWAN1**, which is a LoRaWAN sensor board, part of the kit with the gateway, the **P-NUCLEO-LRWAN2**, and plugged in to a USB port on the Pi.

The purpose of these boards is ostensibly to monitor the office for various factors, including when the temperature gets too high and the number of people occupying said office at any given time. However due to the numerous and varied security issues in almost all modern IoT devices, it has been determined that the device should be audited for security issues.

The reason for the ethernet cable is that LoRaWAN is intended for long distances and, as such, the transmitter and receiver must be kept at a distance, so the signal is not overly saturated, thus rendering the system useless.

The LoPy4 board is a development board that allows users to develop in a variant of Python called MicroPython. In this instance it's acting as part of a temperature regulation system whereby an ambient sensor senses the temperature and reports the data to a web app that displays readings. It also has an LED indicator that will pulse red or blue for a second to indicate the application has received a downlink command.

The second sensor is based on a Nucleo board from STMicroelectronics which is another development board that is programmable in C and allows for firmware to be placed on a device. The LoRaWAN radio interface is provided by a shield board containing a LoRaWAN modem module. There is a PIR proximity sensor wired to the ST board.

Objectives

The following is a list of objectives given to the tester by the CENSIS organisation.

- Gain remote access to the device and get a shell
- Disconnect the device from the internet and set it to the correct time
- Determine what services are running on the device that can be accessed by a web browser (LoRaWAN Network Server, interfaces to monitor data, etc.)
- Crack passwords for relevant services if secured, get into LoRaWAN admin page
- Determine how the network server converts LoRaWAN packets to JSON messages
- Determine how messages are being routed and perform a MITM attack
- Obtain source code running on sensors
- Try and learn about the state of the office without access to network server and keys
- Exploit the device by using factory defaults.

OPENING STAGE

In this stage information is gathered on the system that the tester is attempting to attack. In this instance it is wise to research documentation for each of the components to determine whether there are any known vulnerabilities for the devices and discover how they work for the purposes of further assessment.

To recap, the kit provided has two boxes

- Box 0:
    - NUCLEO-F746ZG
    - LRWAN_GS_HF1 – LoRaWan Gateway Device
    - 868-915MHz Antenna
- Box 1:
    - Raspberry Pi (Indetermined version)
    - Pycom devices (Temp Control):
        - LoPy4 – Dev Board, MicroPython
        - Pysense
    - I-NUCLEO-LRWAN1 – LoRaWAN sensor board with PIR Proximity Sensor
    - Shield Board with LoRaWAN Modem Module

Upon plugging the device in, the tester discovered that it emitted a Wi-Fi signal. This network's SSID was IoT-Demo-6 and it had WPA2-PSK (Wi-Fi Protected Access 2 – Pre-Shared Key) security, which is a security certification intended for personal usage (Beal, 2008). This security scheme makes use of a plain English passphrase as opposed to an encryption key, and as a result it was determined that the best course of action was to attempt to crack the Wi-Fi signal and gain access to the internal network by brute forcing the password.

WIFI CRACKING

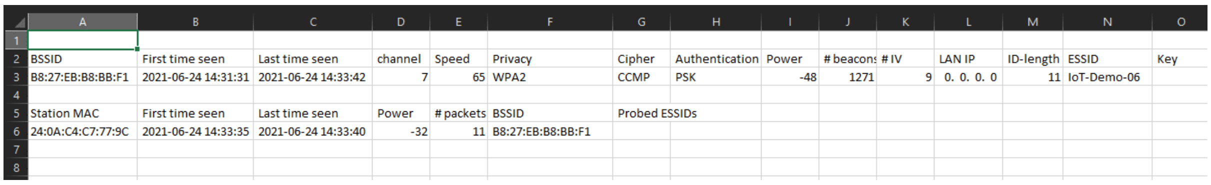
With access to a comprehensive wordlist and some basic skill and tools, cracking the Wi-Fi credentials for this network was relatively trivial. The aircrack-ng suite of tools was used almost exclusively in this stage due to its comprehensive capability in this area.

Capturing handshake

As the tester had no access to specialised hardware to sniff data from external networks and gain access to the Wi-Fi handshake, the process by which an encryption key is generated (WiFi Professionals, 2019), they had to capture then handshake themselves.

In addition, due to internal hardware constraints, specifically the fact that the tester's laptop's internal network card was unable to function in monitor mode, the tester had to make use of a Digitazz brand external USB network adapter, which is why there is a non-standard Wi-Fi interface used in this instance.

The tester initially ran the `iwconfig` command to discover the name of the interface, "wlp0s20f0u3". After using `airmon-ng` to kill processes that would get in the way of accessing the network, the interface named previously was started in monitor mode by the command `sudo airmon-ng start wlp0s20f0u3`. This generated a new interface called "`wlp0s20f0u3mon`" which was subsequently used to capture the BSSID (MAC address) of the device through the second `aircrack-ng` tool, `airodump-ng`.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1															
2	BSSID	First time seen	Last time seen	channel	Speed	Privacy	Cipher	Authentication	Power	# beacons	# IV	LAN IP	ID-length	ESSID	Key
3	B8:27:EB:B8:BB:F1	2021-06-24 14:31:31	2021-06-24 14:33:42	7	65	WPA2	CCMP	PSK	-48	1271	9	0. 0. 0. 0	11	IoT-Demo-06	
4															
5	Station MAC	First time seen	Last time seen	Power	# packets	BSSID	Probed ESSIDs								
6	24:0A:C4:C7:77:9C	2021-06-24 14:33:35	2021-06-24 14:33:40	-32	11	B8:27:EB:B8:BB:F1									
7															
8															

Figure 1, the output in a csv file

The command `sudo airodump-ng -c 7 --bssid B8:27:EB:B8:BB:F1 -w WPAcrack wlp0s20f0u3mon --ignore-negative-one` was ran against the network. The `-c` flag determines the channel of the wireless network, which (as seen in Figure 1) is 7. The `--bssid` flag tells the program what the BSSID is, which is the MAC address for the Wireless Access Point. The `-w` flag determines the file name prefix for the file which will contain the auth handshake, as well as any other output files. `wlp0s20f0u3mon` is, as mentioned previously, the wireless interfaces, and `--ignore-negative-one` fixes an error message that may be returned.

Through running this command, the file `WPA2crack-01.cap` was generated, which contained the wireless handshake.

Gaining Access

After gaining the handshake, the remainder of the process was relatively trivial. The command `aircrack-ng -w words_small.txt -b B8:27:EB:B8:BB:F1 WPA2crack-01.cap` was ran. `Aircrack-ng` is the industry standard network password cracking program, `-w` specifies the location of a wordlist for a dictionary attack (this wordlist was sourced from previous University work), `-b` specifies the BSSID to attack, which was acquired earlier, and `WPA2crack.cap` is the handshake required to access.

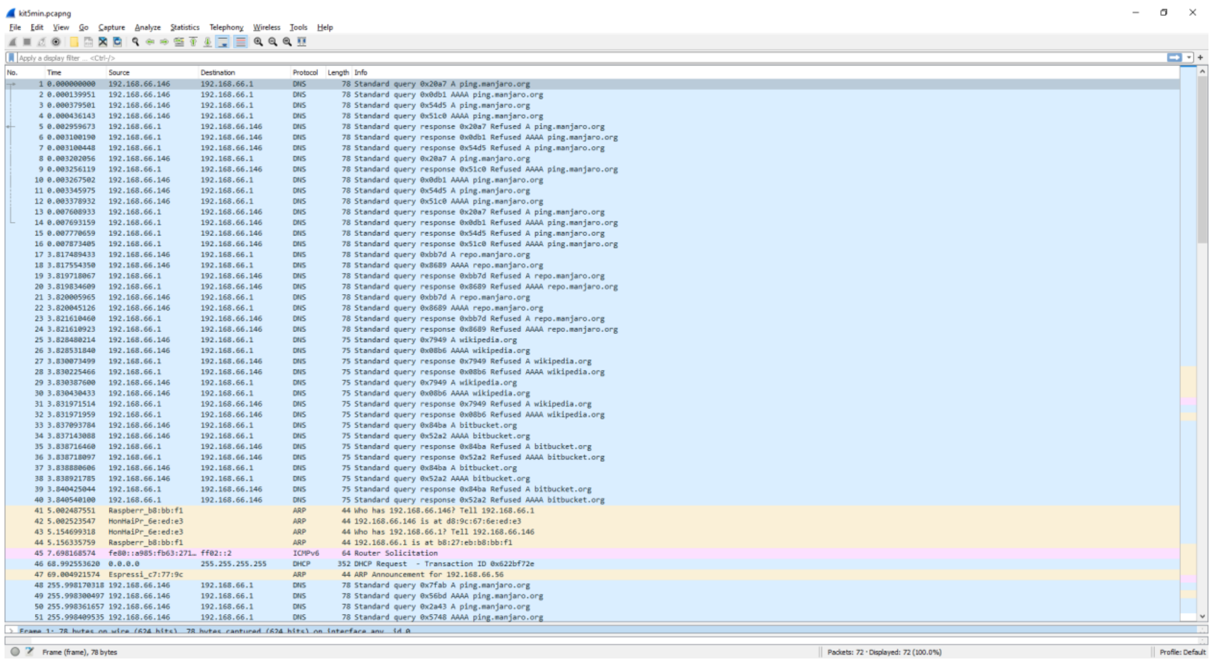
After approximately 1 second, the program found the password to the device, which was “`valegorov`”. This password was tested successfully against the device and as such the tester was able to gain access to the internal device network.

SCANNING

Now that access has been gained to the internal network of the device, the scanning procedure can begin. This stage of the test is primarily concerned with what services are running on the device and how they may be used to further exploit the kit.

Wireshark

The first step taken in the scanning stage was to begin a packet capture through Wireshark. This packet capture was running for approximately 5 minutes, in which time 72 packets were sent and received over the network. The entirety of these packets was between devices with the IP addresses 192.168.66.1 and 192.168.66.146. Surface-level analysis of this capture reveals that the device ending in 1 is the "router" (i.e. the raspberry pi device through which the entirety of the device is run), and the other device is the laptop that was testing, as shown by the repeated refused attempts by the device to connect to various websites (including wikipedia and manjaro.org).


kit5min.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time         Source                 Destination      Protocol Length Info
   1 0.000000000  192.168.66.146         192.168.66.1     DNS       78 Standard query 0x20a7 A ping.manjaro.org
   2 0.000139951  192.168.66.146         192.168.66.1     DNS       78 Standard query 0x0db1 AAAA ping.manjaro.org
   3 0.000379501  192.168.66.146         192.168.66.1     DNS       78 Standard query 0x54d5 A ping.manjaro.org
   4 0.000436143  192.168.66.146         192.168.66.1     DNS       78 Standard query 0x51c0 AAAA ping.manjaro.org
   5 0.002959673  192.168.66.1           192.168.66.146   DNS       78 Standard query response 0x20a7 Refused A ping.manjaro.org
   6 0.003100190  192.168.66.1           192.168.66.146   DNS       78 Standard query response 0x0db1 Refused AAAA ping.manjaro.org
   7 0.003100448  192.168.66.1           192.168.66.146   DNS       78 Standard query response 0x54d5 Refused A ping.manjaro.org
   8 0.003202056  192.168.66.146         192.168.66.1     DNS       78 Standard query 0x20a7 A ping.manjaro.org
   9 0.003256119  192.168.66.1           192.168.66.146   DNS       78 Standard query response 0x51c0 Refused AAAA ping.manjaro.org
  10 0.003267502  192.168.66.146         192.168.66.1     DNS       78 Standard query 0x0db1 AAAA ping.manjaro.org
  11 0.003345975  192.168.66.146         192.168.66.1     DNS       78 Standard query 0x54d5 A ping.manjaro.org
  12 0.003378932  192.168.66.146         192.168.66.1     DNS       78 Standard query 0x51c0 AAAA ping.manjaro.org
  13 0.007608933  192.168.66.1           192.168.66.146   DNS       78 Standard query response 0x20a7 Refused A ping.manjaro.org
  14 0.007693159  192.168.66.1           192.168.66.146   DNS       78 Standard query response 0x0db1 Refused AAAA ping.manjaro.org
  15 0.007770659  192.168.66.1           192.168.66.146   DNS       78 Standard query response 0x54d5 Refused A ping.manjaro.org
  16 0.007873405  192.168.66.1           192.168.66.146   DNS       78 Standard query response 0x51c0 Refused AAAA ping.manjaro.org
  17 3.817489433  192.168.66.146         192.168.66.1     DNS       78 Standard query 0xbb7d A repo.manjaro.org
  18 3.817554350  192.168.66.146         192.168.66.1     DNS       78 Standard query 0x8689 AAAA repo.manjaro.org
  19 3.819718067  192.168.66.1           192.168.66.146   DNS       78 Standard query response 0xbb7d Refused A repo.manjaro.org
  20 3.819834609  192.168.66.1           192.168.66.146   DNS       78 Standard query response 0x8689 Refused AAAA repo.manjaro.org
  21 3.820005965  192.168.66.146         192.168.66.1     DNS       78 Standard query 0xbb7d A repo.manjaro.org
  22 3.820045126  192.168.66.146         192.168.66.1     DNS       78 Standard query 0x8689 AAAA repo.manjaro.org
  23 3.821610460  192.168.66.1           192.168.66.146   DNS       78 Standard query response 0xbb7d Refused A repo.manjaro.org
  24 3.821610923  192.168.66.1           192.168.66.146   DNS       78 Standard query response 0x8689 Refused AAAA repo.manjaro.org
  25 3.828480214  192.168.66.146         192.168.66.1     DNS       75 Standard query 0x7949 A wikipedia.org
  26 3.828531840  192.168.66.146         192.168.66.1     DNS       75 Standard query 0x08b6 AAAA wikipedia.org
  27 3.830073499  192.168.66.1           192.168.66.146   DNS       75 Standard query response 0x7949 Refused A wikipedia.org
  28 3.830225466  192.168.66.1           192.168.66.146   DNS       75 Standard query response 0x08b6 Refused AAAA wikipedia.org
  29 3.830387600  192.168.66.146         192.168.66.1     DNS       75 Standard query 0x7949 A wikipedia.org
  30 3.830430433  192.168.66.146         192.168.66.1     DNS       75 Standard query 0x08b6 AAAA wikipedia.org
  31 3.831971514  192.168.66.1           192.168.66.146   DNS       75 Standard query response 0x7949 Refused A wikipedia.org
  32 3.831971959  192.168.66.1           192.168.66.146   DNS       75 Standard query response 0x08b6 Refused AAAA wikipedia.org
  33 3.837093784  192.168.66.146         192.168.66.1     DNS       75 Standard query 0x84ba A bitbucket.org
  34 3.837143088  192.168.66.146         192.168.66.1     DNS       75 Standard query 0x52a2 AAAA bitbucket.org
  35 3.838716460  192.168.66.1           192.168.66.146   DNS       75 Standard query response 0x84ba Refused A bitbucket.org
  36 3.838718097  192.168.66.1           192.168.66.146   DNS       75 Standard query response 0x52a2 Refused AAAA bitbucket.org
  37 3.838880606  192.168.66.146         192.168.66.1     DNS       75 Standard query 0x84ba A bitbucket.org
  38 3.838921785  192.168.66.146         192.168.66.1     DNS       75 Standard query 0x52a2 AAAA bitbucket.org
  39 3.840425044  192.168.66.1           192.168.66.146   DNS       75 Standard query response 0x84ba Refused A bitbucket.org
  40 3.840540100  192.168.66.1           192.168.66.146   DNS       75 Standard query response 0x52a2 Refused AAAA bitbucket.org
  41 5.002487551  Raspberr_b8:bb:f1                       ARP       44 Who has 192.168.66.146? Tell 192.168.66.1
  42 5.002523547  HonHaiPr_6e:ed:e3                       ARP       44 192.168.66.146 is at d8:9c:67:6e:ed:e3
  43 5.154699318  HonHaiPr_6e:ed:e3                       ARP       44 Who has 192.168.66.1? Tell 192.168.66.146
  44 5.156335759  Raspberr_b8:bb:f1                       ARP       44 192.168.66.1 is at b8:27:eb:b8:bb:f1
  45 7.698168574  fe80::a985:fb63:271… ff02::2          ICMPv6    64 Router Solicitation
  46 68.992553620 0.0.0.0                255.255.255.255  DHCP     352 DHCP Request  - Transaction ID 0x622bf72e
  47 69.004921574 Espressi_c7:77:9c                       ARP       44 ARP Announcement for 192.168.66.56
  48 255.998170318 192.168.66.146        192.168.66.1     DNS       78 Standard query 0x7fab A ping.manjaro.org
  49 255.998300497 192.168.66.146        192.168.66.1     DNS       78 Standard query 0x56bd AAAA ping.manjaro.org
  50 255.998361657 192.168.66.146        192.168.66.1     DNS       78 Standard query 0x2a43 A ping.manjaro.org
  51 255.998409535 192.168.66.146        192.168.66.1     DNS       78 Standard query 0x5748 AAAA ping.manjaro.org

> Frame 1: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface any, id 0

Frame (frame), 78 bytes                    Packets: 72 · Displayed: 72 (100.0%)          Profile: Default
```

Figure 2, packets 1-51

Nmap

Now that the tester had gained the target's IP address, they could run an nmap scan against it. Nmap, or network mapper, is the industry standard tool for scanning and mapping networks that are connected to the internet. The main use of this tool is to reveal any ports open to the internet, and, as such, reveal the services running on these ports.

A standard nmap scan (`nmap 192.168.66.1`) revealed that three ports were open, port 22 (SSH), port 53 (a domain server), and port 8080 (an http proxy, used for websites).

All three of these ports ran on TCP, or the Transmission Control Protocol. As a result of this, the next scan ran against the IP address targeted these ports exclusively. Additionally, the scan was opened to all ports (from 1-65535) to account for non-standard port usage, and the double verbose flag was added to return as much useful information as possible. This scan (`nmap -sT -p- -vv -T5 192.168.66.1`) showed 2 additional known ports in use, 1993 running MQTT, a protocol used in IoT devices to send simple messages between devices (Random Nerd Tutorials, n.d.), and 4369 running EPMD, the Erlang Port Mapper Daemon, a program acting as a name server for "hosts involved in distributed Erlang computations" (Ericsson AB, n.d.). An extra 5 unknown ports were open.

A similar, final nmap scan was run, with the only difference being the addition of the `-A` flag, which "Enables OS detection, version detection, script scanning, and traceroute" (Lyon, 2009). From this, further information was gained, such as the precise version numbers of the software, the OS running on the device (Raspbian), and a further two extra http servers hosted on ports 7245 and 8245, to which the tester shall return later.

CRACKING SSH

The first port the tester discovered was open during the scanning phase was SSH. SSH, or the Secure Shell protocol, allows for terminal-based access to an external computer device through a connection established on a host machine using the remote device's username and IP address, as well as a password if one is present.

Since this device is a simple IoT controller, the tester presumed the device's username will have remained default, on Raspbian the default username is pi, so with this assumption in addition to the IP address gathered in the previous stage, the tester made use of the Hydra password cracking tool and the previously used words_small.txt wordlist to attempt to crack the password using a simple dictionary attack.

After running `hydra -l pi -P words_small.txt ssh://192.168.66.1` for approximately 30 minutes, valid credentials were gathered, username: pi, password: sourcetec. After logging in using `ssh pi@192.168.66.1` and a valid password, it was revealed that the pi user was a sudoer, and as such a root shell was accessible via the `sudo bash` command, therefore all aspects of the filesystem were now freely accessible to the hacker.

Later during a subsequent stage, another account was found named "developer", this process was repeated but no valid credentials were discovered.

This stage was the one in which the tester set the clock to the correct time, as per the objectives list. This was achieved by turning off ntp synchronisation (`sudo systemctl stop ntp`), which was wildly inaccurate due to the fact the device was not connected to the internet and thus not connected to an ntp server. After this it was a simple case of setting the time manually, `sudo timedatectl set-time hh:mm:ss` where hh mm and ss was the hour, minute, and second that it was at the time.

ENUMERATION AND EXFILTRATION

Opening Enumeration

The first step of the enumeration phase in this instance was to gain access to a script that ran basic enumeration functionality on the remote device, to gain an overview of the system, where it may be vulnerable or exploitable, and to check the access the tester already had with the pi user.

The tester's preferred tool for this is a shell script called LSE, or Linux smart Enumeration, developed by Diego Treitos (Treitos, 2019). They acquired this by running `curl "https://github.com/diego-treitos/linux-smartenumeration/raw/master/lse.sh" -Lo lse.sh;chmod 700 lse.sh` which downloaded the script and allowed it to be executed, and then `scp -r ~/Downloads/lse pi@192.168.66.1:~/Downloads/lse` to get the containing folder to the remote device in order to execute it. The full results of this scan are available in Appendix A.

Much of the information returned in this scan is things the tester already knew, for example, that the user has access to admin root privileges and that they could read subdirectories under /home, however other interesting information was made available for the first time thanks to this scan, that there are other users on the device (although they are not in the sudoers group), that sudoing is possible without a password, and that there are processes running on that user's end.

Dirbuster

Dirbuster is a GUI directory enumeration tool with dirb serving as the backend. This tool allows a user to view the full URL (with subdirectories) of a service running on a given website given the root directory and a wordlist of common directory names.

The tester ran the tool against three ports that they had determined previously were running web servers (7245, 8080, and 8245). The tester managed to gain un-protected access to the webpages on ports 7245 and 8245, both of which were determined to be frontends for the temperature sensor's reported results, from this we can infer that the code running the backend is on the device somewhere. Dirbuster found no extra subdirectories under the root directory on either of these pages

Port 8080's page, when accessed from a browser, immediately displayed a JavaScript popup window prompting the user to input admin credentials, however due to time and resource constraints placed on the tester, they were unable to crack this and gain access to it. Further information on this will be supplied in the "Further Work" section.

The HTML the tester managed to exfiltrate is in Appendix B, and the data from the three dirbuster attempts is in Appendix A.

Nikto

Nikto is another command line-based tool that scans web servers for known vulnerabilities. In this instance the tester pointed the scan to the site at port 8080 exclusively to find a method of entry for the admin page located at that port. The results of the scan were relatively inconclusive; however, three vulnerabilities were discovered.

The first vulnerability was "`The anti-clickjacking X-Frame-Options header is not present`", this issue is irrelevant on this device due to the fact that a clickjacking attack, being a "malicious technique of tricking a Web user into clicking on something different from what the user perceives they are clicking on" (Acunetix, n.d.), is only effective on larger scales to public facing websites because of its wide attack surface, which this site is explicitly not able to have.

The second vuln the Nikto scan said "`The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS`". This vulnerability could allow an attacker to gain access to the backend of a website through malicious code injection, however this is not a viable attack vector in this instance due to the fact the backend is accessible through SSH.

The final vulnerability was "`The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type`", which is yet another XSS attack surface that is similarly non-viable for the reasons listed above.

To conclude, the Nikto scan was not particularly helpful in this case, however the issues on here may need to be addressed if the scale of the rollout of these devices increases.

Exfiltration

This final stage of discovery on the device itself is mainly concerned with getting files that may be of interest from the client device, being the raspberry pi, to the host, in this case a Laptop running Manjaro KDE Linux.

One of the KDE desktop environment's file managers, Dolphin, has inbuilt support for FISH, the "Files Transferred Over Shell protocol), which allows files to be transferred from one device to another through GUI interaction, this made passing files between the two devices trivial.

The files exfiltrated from the device were divided into roughly three categories: files from the pi user's home directory, files from the developer user's home directory, and files generated from commands executed on the device.

The program/user generated files are the smallest category, three commands were run during this stage. These commands were `tree`, `cd /; tree`, and `cd ~; ps -aux` in that order. The tree command generates a tree of all directories, subdirectories, and files from a certain location, in this instance the tester first ran it from the pi user's home directory to find any files possibly relating to the task at hand, then from the root directory to create a list of all files running on the machine. This approach generated a file far too big to analyse on a large scale, so it was mostly used to search for files under `/home/developer/`, which, as we shall see later, was worthwhile.

The `ps -aux` command simply lists processes from all users, with the user or owner listed in the output, and processes that haven't been executed from terminal. The output of this file is in Appendix A and reveals all the processes running locally, including the Mosquitto and epmd servers spoken about earlier, as well as LoRaWAN and root processes that give an insight into the state of the device.

Next, the pi user. Nothing out of the ordinary was found here, the account uses chromium for testing and as such has logs that reflect this, however there was nothing of note the tester could locate here that fit the use case

The developer user, however, contains the entire source code for the temperature sensor, as well as an interesting readme.txt file that links to /etc/system/system, all of which was exfiltrated and is available in Appendix B. Analysis of these files will occur in the following section.

POST-ACCESS ANALYSIS

This section is regarding the analysis of the files retrieved from the device, with reference to the source code behind the website hosted on the device that was hacked. The first file read from this section of the filesystem was a `readme.txt` file (Appendix B) intended for a user called Norman from an individual called Kenny, this file is intended to inform Norman how to set up the relevant software for the temperature sensor device from scratch. The information found in this file can inform the tester exactly where to look and what the dependencies are for the device and therefore gain a full understanding of how to proceed.

Given further access, it would be possible to alter the contents of the python files arbitrarily, therefore giving either subtly or wildly inaccurate data points, or to change the purpose of the file entirely.

In addition to this, the “system” files recovered from the device give significant amounts of information on the services running on the device, including most notably the system file path to said services, a malicious actor with enough skill could subtly alter the behaviour of the programs found in these file paths (for example, Msoquitto), to change their behaviour arbitrarily. This could include injecting code that results in a man in the middle attack being feasible and used in tandem with access to the python files as mentioned previously, the actor could be able to collect a significant amount of data on the network on which the device is hosted.

CONCLUSION

In conclusion, the exploitation of this device in most of the ways specified by the client was trivial. Gaining Remote CLI access to the system and altering its behaviour (in this case by correcting the system time) can be accomplished on the system as it is with very little effort, and given access to the system for a longer amount of time than was allotted could very well result in a full exploitation of the system as specified by the client, including the ability to infer information about the office in which the device is hosted, gain full access to the LoRaWAN admin page, and performing a man in the middle attack by discovering how the messages are being routed.

REFERENCES

Beal, V., 2008. *WPA2-PSK.* [Online]
Available at: https://www.webopedia.com/definitions/wpa2-psk/
[Accessed 29 June 2021].

Ericsson AB, n.d. *epmd.* [Online]
Available at: https://erlang.org/doc/man/epmd.html
[Accessed 29 June 2021].

Random Nerd Tutorials, n.d. *What is MQTT and How It Works.* [Online]
Available at: https://randomnerdtutorials.com/what-is-mqtt-and-how-it-works/
[Accessed 29 June 2021].

WiFi Professionals, 2019. *4-WAY HANDSHAKE.* [Online]
Available at: https://www.wifi-professionals.com/2019/01/4-way-handshake
[Accessed 29 June 2021].

APPENDIX

Appendix A – Tool Outputs

LSE

```
---
---

 LSE Version: 3.3

        User: pi
     User ID: 1000
    Password: ******
        Home: /home/pi
        Path:
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games:/usr
/games
       umask: 0022

    Hostname: raspberrypi
       Linux: 5.10.17-v7+
Distribution: Raspbian GNU/Linux 10 (buster)
Architecture: armv7l

==================================================================( users )=====
[i] usr000 Current user groups........................................... yes!
[*] usr010 Is current user in an administrative group?.................... yes!
[*] usr020 Are there other users in administrative groups?................ nope
[*] usr030 Other users with shell......................................... yes!
[i] usr040 Environment information........................................ skip
[i] usr050 Groups for other users......................................... skip
[i] usr060 Other users.................................................... skip
[*] usr070 PATH variables defined inside /etc............................. yes!
[!] usr080 Is '.' in a PATH variable defined inside /etc?................. nope
===================================================================( sudo )=====
[!] sud000 Can we sudo without a password?................................ yes!
---
uid=0(root) gid=0(root) groups=0(root),117(lpadmin)
---
[*] sud040 Can we read sudoers files?..................................... nope
[*] sud050 Do we know if any other users used sudo?....................... nope
============================================================( file system )=====
[*] fst000 Writable files outside user's home............................. yes!
[*] fst010 Binaries with setuid bit....................................... yes!
[!] fst020 Uncommon setuid binaries....................................... yes!
---
```

```
/usr/bin/Xvnc
/usr/bin/vncserver-x11
/usr/bin/gpio
/usr/bin/bwrap
/usr/bin/ntfs-3g
/usr/sbin/mount.cifs
/usr/lib/arm-linux-gnueabihf/gstreamer1.0/gstreamer-1.0/gst-ptp-helper
---
[!] fst030 Can we write to any setuid binary?............................ nope
[*] fst040 Binaries with setgid bit...................................... skip
[!] fst050 Uncommon setgid binaries...................................... skip
[!] fst060 Can we write to any setgid binary?............................ skip
[*] fst070 Can we read /root?............................................ nope
[*] fst080 Can we read subdirectories under /home?....................... yes!
[*] fst090 SSH files in home directories................................. yes!
[*] fst100 Useful binaries............................................... yes!
[*] fst110 Other interesting files in home directories................... nope
[!] fst120 Are there any credentials in fstab/mtab?...................... nope
[*] fst130 Does 'pi' have mail?.......................................... nope
[!] fst140 Can we access other users mail?............................... nope
[*] fst150 Looking for GIT/SVN repositories.............................. nope
[!] fst160 Can we write to critical files?............................... nope
[!] fst170 Can we write to critical directories?......................... nope
[!] fst180 Can we write to directories from PATH defined in /etc?........ nope
[!] fst190 Can we read any backup?....................................... nope
[!] fst200 Are there possible credentials in any shell history file?..... nope
[i] fst500 Files owned by user 'pi'...................................... skip
[i] fst510 SSH files anywhere............................................ skip
[i] fst520 Check hosts.equiv file and its contents....................... skip
[i] fst530 List NFS server shares........................................ skip
[i] fst540 Dump fstab file............................................... skip
=================================================================( system )=====
[i] sys000 Who is logged in.............................................. skip
[i] sys010 Last logged in users.......................................... skip
[!] sys020 Does the /etc/passwd have hashes?............................. nope
[!] sys022 Does the /etc/group have hashes?.............................. nope
[!] sys030 Can we read shadow files?..................................... nope
[*] sys040 Check for other superuser accounts............................ nope
[*] sys050 Can root user log in via SSH?................................. nope
[i] sys060 List available shells......................................... skip
[i] sys070 System umask in /etc/login.defs............................... skip
[i] sys080 System password policies in /etc/login.defs................... skip
===============================================================( security )=====
[*] sec000 Is SELinux present?........................................... nope
[*] sec010 List files with capabilities.................................. yes!
[!] sec020 Can we write to a binary with caps?........................... nope
[!] sec030 Do we have all caps in any binary?............................. nope
[*] sec040 Users with associated capabilities............................. nope
[!] sec050 Does current user have capabilities?........................... skip
[!] sec060 Can we read the auditd log?.................................... nope
========================================================( recurrent tasks )=====
[*] ret000 User crontab................................................... nope
[!] ret010 Cron tasks writable by user.................................... nope
[*] ret020 Cron jobs...................................................... yes!
[*] ret030 Can we read user crontabs...................................... nope
[*] ret040 Can we list other user cron tasks?............................. nope
[*] ret050 Can we write to any paths present in cron jobs................. nope
[!] ret060 Can we write to executable paths present in cron jobs.......... skip
[i] ret400 Cron files..................................................... skip
[*] ret500 User systemd timers............................................ nope
[!] ret510 Can we write in any system timer?.............................. nope
[i] ret900 Systemd timers................................................. skip
================================================================( network )=====
[*] net000 Services listening only on localhost........................... yes!
[!] net010 Can we sniff traffic with tcpdump?............................. nope
[i] net500 NIC and IP information......................................... skip
[i] net510 Routing table.................................................. skip
[i] net520 ARP table...................................................... skip
[i] net530 Nameservers.................................................... skip
[i] net540 Systemd Nameservers............................................ skip
[i] net550 Listening TCP.................................................. skip
[i] net560 Listening UDP.................................................. skip
===============================================================( services )=====
[!] srv000 Can we write in service files?................................. nope
[!] srv010 Can we write in binaries executed by services?................. nope
[*] srv020 Files in /etc/init.d/ not belonging to root.................... nope
[*] srv030 Files in /etc/rc.d/init.d not belonging to root................ nope
[*] srv040 Upstart files not belonging to root............................ nope
[*] srv050 Files in /usr/local/etc/rc.d not belonging to root............. nope
[i] srv400 Contents of /etc/inetd.conf.................................... skip
[i] srv410 Contents of /etc/xinetd.conf................................... skip
[i] srv420 List /etc/xinetd.d if used..................................... skip
[i] srv430 List /etc/init.d/ permissions.................................. skip
[i] srv440 List /etc/rc.d/init.d permissions.............................. skip
[i] srv450 List /usr/local/etc/rc.d permissions........................... skip
[i] srv460 List /etc/init/ permissions.................................... skip
[!] srv500 Can we write in systemd service files?......................... nope
[!] srv510 Can we write in binaries executed by systemd services?......... nope
[*] srv520 Systemd files not belonging to root............................ nope
[i] srv900 Systemd config files permissions............................... skip
===============================================================( software )=====
[!] sof000 Can we connect to MySQL with root/root credentials?............ nope
[!] sof010 Can we connect to MySQL as root without password?............... nope
[!] sof015 Are there credentials in mysql_history file?.................... nope
[!] sof020 Can we connect to PostgreSQL template0 as postgres and no pass?. nope
[!] sof020 Can we connect to PostgreSQL template1 as postgres and no pass?. nope
[!] sof020 Can we connect to PostgreSQL template0 as psql and no pass?..... nope
[!] sof020 Can we connect to PostgreSQL template1 as psql and no pass?..... nope
[*] sof030 Installed apache modules........................................ nope
[!] sof040 Found any .htpasswd files?...................................... nope
[!] sof050 Are there private keys in ssh-agent?............................ nope
[!] sof060 Are there gpg keys cached in gpg-agent?......................... nope
[!] sof070 Can we write to a ssh-agent socket?............................. nope
[!] sof080 Can we write to a gpg-agent socket?............................. yes!
---
/run/user/1000/gnupg/S.gpg-agent.browser
/run/user/1000/gnupg/S.gpg-agent
/run/user/1000/gnupg/S.gpg-agent.ssh
/run/user/1000/gnupg/S.gpg-agent.extra
---
[i] sof500 Sudo version.................................................... skip
[i] sof510 MySQL version................................................... skip
[i] sof520 Postgres version................................................ skip
[i] sof530 Apache version.................................................. skip
============================================================( containers )=====
[*] ctn000 Are we in a docker container?................................... nope
[*] ctn010 Is docker available?............................................ nope
[!] ctn020 Is the user a member of the 'docker' group?..................... nope
[*] ctn200 Are we in a lxc container?...................................... nope
[!] ctn210 Is the user a member of any lxc/lxd group?...................... nope
=============================================================( processes )=====
[i] pro000 Waiting for the process monitor to finish....................... yes!
[i] pro001 Retrieving process binaries..................................... yes!
[i] pro002 Retrieving process users........................................ yes!
[!] pro010 Can we write in any process binary?............................. nope
[*] pro020 Processes running with root permissions......................... yes!
[*] pro030 Processes running by non-root users with shell.................. yes!
[i] pro500 Running processes............................................... skip
[i] pro510 Running process binaries and permissions........................ skip

==================================( FINISHED )==================================
```

DIRBUSTER

Port 7425

```
DirBuster 1.0-RC1 - Report
http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project
Report produced on Fri Jun 25 21:31:46 BST 2021
```

```
--------------------------------

http://192.168.66.1:7245
--------------------------------
Directories found during testing:

Dirs found with a 200 response:

/

Dirs found with a 301 response:

//


--------------------------------
--------------------------------
```

Port 8080

```
DirBuster 1.0-RC1 - Report
http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project
Report produced on Fri Jun 25 21:26:20 BST 2021
--------------------------------

http://192.168.66.1:8080
--------------------------------
Directories found during testing:

Dirs found with a 301 response:

/

Dirs found with a 401 response:

/admin/
/favicon.ico/

Dirs found with a 405 response:

/admin/timeline/


--------------------------------
--------------------------------
```

Port 8245

```
DirBuster 1.0-RC1 - Report
http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project
Report produced on Fri Jun 25 21:35:25 BST 2021
--------------------------------

http://192.168.66.1:8245
--------------------------------
Directories found during testing:

Dirs found with a 200 response:

/

Dirs found with a 301 response:

//



--------------------------------
--------------------------------
```

PS -AUX

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.4  0.8  33764  8228 ?        Rs   00:16   0:09 /sbin/init splash
root         2  0.0  0.0      0     0 ?        S    00:16   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        I<   00:16   0:00 [rcu_gp]
root         4  0.0  0.0      0     0 ?        I<   00:16   0:00 [rcu_par_gp]
root         6  0.0  0.0      0     0 ?        I<   00:16   0:00 [kworker/0:0H-
mmc_complete]
root         8  0.0  0.0      0     0 ?        I<   00:16   0:00 [mm_percpu_wq]
root         9  0.0  0.0      0     0 ?        S    00:16   0:00 [rcu_tasks_rude_]
root        10  0.0  0.0      0     0 ?        S    00:16   0:00 [rcu_tasks_trace]
root        11  0.0  0.0      0     0 ?        S    00:16   0:00 [ksoftirqd/0]
root        12  0.0  0.0      0     0 ?        I    00:16   0:00 [rcu_sched]
root        13  0.0  0.0      0     0 ?        S    00:16   0:00 [migration/0]
root        14  0.0  0.0      0     0 ?        S    00:16   0:00 [cpuhp/0]
root        15  0.0  0.0      0     0 ?        S    00:16   0:00 [cpuhp/1]
root        16  0.0  0.0      0     0 ?        S    00:16   0:00 [migration/1]
root        17  0.0  0.0      0     0 ?        S    00:16   0:00 [ksoftirqd/1]
root        20  0.0  0.0      0     0 ?        S    00:16   0:00 [cpuhp/2]
root        21  0.0  0.0      0     0 ?        S    00:16   0:00 [migration/2]
root        22  0.0  0.0      0     0 ?        S    00:16   0:00 [ksoftirqd/2]
root        25  0.0  0.0      0     0 ?        S    00:16   0:00 [cpuhp/3]
root        26  0.0  0.0      0     0 ?        S    00:16   0:00 [migration/3]
root        27  0.0  0.0      0     0 ?        S    00:16   0:00 [ksoftirqd/3]
root        30  0.0  0.0      0     0 ?        S    00:16   0:00 [kdevtmpfs]
root        31  0.0  0.0      0     0 ?        I<   00:16   0:00 [netns]
root        35  0.0  0.0      0     0 ?        S    00:16   0:00 [kauditd]
root        36  0.0  0.0      0     0 ?        S    00:16   0:00 [khungtaskd]
root        37  0.0  0.0      0     0 ?        S    00:16   0:00 [oom_reaper]
root        38  0.0  0.0      0     0 ?        I<   00:16   0:00 [writeback]
root        39  0.0  0.0      0     0 ?        S    00:16   0:00 [kcompactd0]
root        57  0.0  0.0      0     0 ?        I<   00:16   0:00 [kblockd]
root        58  0.0  0.0      0     0 ?        I<   00:16   0:00 [blkcg_punt_bio]
root        59  0.0  0.0      0     0 ?        S    00:16   0:00 [watchdogd]
root        61  0.0  0.0      0     0 ?        I<   00:16   0:00 [kworker/2:1H-
kblockd]
root        62  0.0  0.0      0     0 ?        I<   00:16   0:00 [rpciod]
root        63  0.0  0.0      0     0 ?        I<   00:16   0:00 [kworker/u9:0-
hci0]
root        64  0.0  0.0      0     0 ?        I<   00:16   0:00 [xprtiod]
root        65  0.0  0.0      0     0 ?        S    00:16   0:00 [kswapd0]
root        66  0.0  0.0      0     0 ?        I<   00:16   0:00 [nfsiod]
root        67  0.0  0.0      0     0 ?        I<   00:16   0:00 [iscsi_eh]
root        68  0.0  0.0      0     0 ?        I<   00:16   0:00 [iscsi_destroy]
root        69  0.0  0.0      0     0 ?        I<   00:16   0:00 [dwc_otg]
root        70  0.0  0.0      0     0 ?        I<   00:16   0:00 [DWC Notificatio]
root        72  0.0  0.0      0     0 ?        S<   00:16   0:00 [vchiq-slot/0]
root        73  0.0  0.0      0     0 ?        S<   00:16   0:00 [vchiq-recy/0]
root        74  0.0  0.0      0     0 ?        S<   00:16   0:00 [vchiq-sync/0]
root        75  0.0  0.0      0     0 ?        I<   00:16   0:00 [zswap-shrink]
root        77  0.0  0.0      0     0 ?        I<   00:16   0:00 [mmc_complete]
root        78  0.0  0.0      0     0 ?        I<   00:16   0:00 [kworker/1:1H-
kblockd]
root        80  0.0  0.0      0     0 ?        I<   00:16   0:00 [kworker/3:1H-
kblockd]
root        82  0.0  0.0      0     0 ?        S    00:16   0:00 [jbd2/mmcblk0p2-]
root        83  0.0  0.0      0     0 ?        I<   00:16   0:00 [ext4-rsv-conver]
root        84  0.0  0.0      0     0 ?        I<   00:16   0:00 [ipv6_addrconf]
root       102  0.0  0.0      0     0 ?        S    00:16   0:00 [scsi_eh_0]
root       103  0.0  0.0      0     0 ?        I<   00:16   0:00 [scsi_tmf_0]
root       104  0.0  0.0      0     0 ?        S    00:16   0:00 [usb-storage]
root       108  0.0  0.0      0     0 ?        S    00:17   0:00 [irq/199-usb-001]
root       111  0.0  0.7  33252  7452 ?        Ss   00:17   0:01
/lib/systemd/systemd-journald
root       142  0.0  0.0      0     0 ?        I    00:17   0:00 [kworker/2:2-
events]
root       153  0.0  0.4  18592  3904 ?        Ss   00:17   0:01
/lib/systemd/systemd-udevd
root       176  0.0  0.0      0     0 ?        S    00:17   0:00 [vchiq-keep/0]
root       178  0.0  0.0      0     0 ?        S<   00:17   0:00 [SMIO]
root       194  0.0  0.0      0     0 ?        I<   00:17   0:00 [mmal-vchiq]
root       200  0.0  0.0      0     0 ?        I<   00:17   0:00 [mmal-vchiq]
root       202  0.0  0.0      0     0 ?        I<   00:17   0:00 [mmal-vchiq]
root       204  0.0  0.0      0     0 ?        I<   00:17   0:00 [mmal-vchiq]
root       227  0.0  0.0      0     0 ?        I<   00:17   0:00 [cfg80211]
root       229  0.0  0.0      0     0 ?        I<   00:17   0:00 [uas]
root       233  0.0  0.0      0     0 ?        I<   00:17   0:00 [brcmf_wq/mmc1:0]
root       234  0.0  0.0      0     0 ?        S    00:17   0:00 [brcmf_wdog/mmc1]
systemd+   306  0.0  0.3  22416  2896 ?        Ssl  00:17   0:01
/lib/systemd/systemd-timesyncd
message+   343  0.0  0.3   6708  3500 ?        Ss   00:17   0:00 /usr/bin/dbus-
daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --
syslog-only
nobody     346  0.0  0.2   4320  2016 ?        Ss   00:17   0:00 /usr/sbin/thd --
triggers /etc/triggerhappy/triggers.d/ --socket /run/thd.socket --user nobody --
deviceglob /dev/input/event*
root       347  0.0  0.2   7948  2196 ?        Ss   00:17   0:00 /usr/sbin/cron -f
root       351  0.0  1.1  65076 10428 ?        Ssl  00:17   0:00
/usr/lib/udisks2/udisksd
root       352  0.0  0.0  27656    80 ?        SLsl 00:17   0:00 /usr/sbin/rngd -r
/dev/hwrng
root       359  0.0  0.4  11768  4492 ?        SNs  00:17   0:00 /usr/sbin/alsactl
-E HOME=/run/alsa -s -n 19 -c rdaemon
root       365  0.1  2.8  44116 27068 ?        Ssl  00:17   0:03 /usr/bin/python
/usr/local/bin/temperature.py
root       373  0.0  0.6  13044  5772 ?        Ss   00:17   0:00
/lib/systemd/systemd-logind
root       377  0.1  1.6  32628 16072 ?        Ssl  00:17   0:02 /usr/bin/python
/usr/local/bin/pir.py
root       378  0.0  1.5  22292 15088 ?        Ss   00:17   0:01 /usr/bin/python
/usr/local/bin/debug_output.py
root       385  0.0  1.6  22292 15172 ?        Ss   00:17   0:01 /usr/bin/python
/usr/local/bin/debug_output_pir.py
avahi      398  0.0  0.2   5768  2528 ?        Ss   00:17   0:00 avahi-daemon:
running [raspberrypi.local]
root       405  0.0  0.2  25512  2768 ?        Ssl  00:17   0:00
/usr/sbin/rsyslogd -n -iNONE
root       407  0.0  0.6  27640  6416 ?        Ss   00:17   0:00 /usr/sbin/cupsd -
l
avahi      427  0.0  0.0   5768   252 ?        S    00:17   0:00 avahi-daemon:
chroot helper
root       429  0.0  0.1   2904  1840 ?        Ss   00:17   0:00 /sbin/dhcpcd -q -
b
mosquit+   438  0.0  0.4   8812  4688 ?        Ss   00:17   0:01
/usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf
```

```
lorawan    439  0.0  0.0   1940   368 ?        Ss   00:17   0:00 /bin/sh
/usr/lib/lorawan-server/bin/lorawan-server
epmd       441  0.0  0.1   4876  1040 ?        Ss   00:17   0:00 /usr/bin/epmd -
systemd
root       462  0.0  0.9  40496  8588 ?        Ssl  00:17   0:00 /usr/sbin/cups-
browsed
root       471  0.0  0.7  39992  7284 ?        Ssl  00:17   0:00 /usr/sbin/lightdm
root       473  0.0  0.7  40456  7308 ?        Ssl  00:17   0:00
/usr/lib/policykit-1/polkitd --no-debug
lorawan    476  0.6  2.6  82156 25016 ?        Sl   00:17   0:14
/usr/lib/erlang/erts-10.2.4/bin/beam.smp -Bd -- -root /usr/lib/erlang -progname
erl -- -home /var/lib/lorawan-server -- -noshell -noinput -sname lorawan -pa
/usr/lib/lorawan-server/lib/amqp_client-3.6.14/ebin /usr/lib/lorawan-
server/lib/cowboy-2.5.0/ebin /usr/lib/lorawan-server/lib/cowlib-2.6.0/ebin
/usr/lib/lorawan-server/lib/eid-0.5.0/ebin /usr/lib/lorawan-server/lib/emqttc-
0.8.0/ebin /usr/lib/lorawan-server/lib/erlmongo-0.2/ebin /usr/lib/lorawan-
server/lib/gen_smtp-0.13.0/ebin /usr/lib/lorawan-server/lib/goldrush-0.1.9/ebin
/usr/lib/lorawan-server/lib/gun-1.3.0/ebin /usr/lib/lorawan-server/lib/iso8601-
1.3.1/ebin /usr/lib/lorawan-server/lib/jsx-2.9.0/ebin /usr/lib/lorawan-
server/lib/lager-3.6.8/ebin /usr/lib/lorawan-server/lib/lorawan_server-0.6.7/ebin
/usr/lib/lorawan-server/lib/rabbit_common-3.6.14/ebin /usr/lib/lorawan-
server/lib/ranch-1.6.2/ebin /usr/lib/lorawan-server/lib/recon-2.3.2/ebin -s
lorawan_app -lager log_root "/var/log/lorawan-server" -config /usr/lib/lorawan-
server/releases/0.6.7/sys.config
root       494  0.0  0.5  10728  5308 ?        Ss   00:17   0:00 /usr/sbin/sshd -D
dnsmasq    497  0.0  0.1  11080  1864 ?        S    00:17   0:01 /usr/sbin/dnsmasq
-x /run/dnsmasq/dnsmasq.pid -u dnsmasq -r /run/dnsmasq/resolv.conf -7
/etc/dnsmasq.d,.dpkg-dist,.dpkg-old,.dpkg-new --local-service --trust-
anchor=.,20326,8,2,e06d44b80b8f1d39a95c0b0d7c65d08458e880409bbc683457104237c7f8ec8
d
root       498  0.0  0.2   7480  2000 ?        Ss   00:17   0:00 /usr/sbin/hostapd
-B -P /run/hostapd.pid -B /etc/hostapd/hostapd.conf
lorawan    529  0.0  0.0   1860   336 ?        Ss   00:17   0:00 erl_child_setup
1024
lp         534  0.0  0.5  13832  4740 ?        S    00:17   0:00
/usr/lib/cups/notifier/dbus dbus://
lp         535  0.0  0.4  13832  4544 ?        S    00:17   0:00
/usr/lib/cups/notifier/dbus dbus://
lp         536  0.0  0.4  13832  4708 ?        S    00:17   0:00
/usr/lib/cups/notifier/dbus dbus://
lp         537  0.0  0.5  13832  4792 ?        S    00:17   0:00
/usr/lib/cups/notifier/dbus dbus://
root       568  0.5  3.7 200508 35868 tty7     Ssl+ 00:17   0:12
/usr/lib/xorg/Xorg :0 -seat seat0 -auth /var/run/lightdm/root/:0 -nolisten tcp vt7
-novtswitch
```

```
root       569  0.0  0.1   4308  1364 tty1     Ss+  00:17   0:00 /sbin/agetty -o -
p -- \u --noclear tty1 linux
lorawan    604  0.0  0.0   1940   412 ?        Ss   00:17   0:00 sh -s disksup
lorawan    614  0.0  0.0   1720   328 ?        Ss   00:17   0:00
/usr/lib/erlang/lib/os_mon-2.4.7/priv/bin/memsup
lorawan    616  0.0  0.0   1852   324 ?        Ss   00:17   0:00
/usr/lib/erlang/lib/os_mon-2.4.7/priv/bin/cpu_sup
root       622  0.0  0.7  32004  7280 ?        Sl   00:17   0:00 lightdm --
session-child 18 21
lightdm    626  0.0  0.7  14704  7384 ?        Ss   00:17   0:02
/lib/systemd/systemd --user
lightdm    627  0.0  0.3  16864  3544 ?        S    00:17   0:00 (sd-pam)
lightdm    637  0.4  4.4 119192 41680 ?        Ssl  00:17   0:09 /usr/sbin/pi-
greeter
lightdm    641  0.0  0.3   6416  3000 ?        Ss   00:17   0:00 /usr/bin/dbus-
daemon --session --address=systemd: --nofork --nopidfile --systemd-activation --
syslog-only
lightdm    642  0.0  0.6  43596  6332 ?        Ssl  00:17   0:00
/usr/lib/gvfs/gvfsd
lightdm    647  0.0  0.6  55732  6608 ?        Sl   00:17   0:00
/usr/lib/gvfs/gvfsd-fuse /run/user/109/gvfs -f -o big_writes
root       658  0.0  0.4  11232  4624 ?        S    00:17   0:00 lightdm --
session-child 14 21
lorawan    667  0.0  0.0   2564   448 ?        Ss   00:17   0:00 inet_gethost 4
lorawan    668  0.0  0.1   2696  1392 ?        S    00:17   0:00 inet_gethost 4
root       680  0.0  0.0   2136   112 ?        S    00:17   0:00
/usr/bin/hciattach /dev/serial1 bcm43xx 3000000 flow - b8:27:eb:47:44:0e
root       682  0.0  0.0      0     0 ?        I<   00:17   0:00 [kworker/u9:1-
hci0]
root       686  0.0  0.3   9536  3308 ?        Ss   00:17   0:00
/usr/lib/bluetooth/bluetoothd
root       714  0.0  0.6  12240  6216 ?        Ss   00:19   0:00 sshd: pi [priv]
pi         717  0.0  0.7  14712  7328 ?        Ss   00:19   0:02
/lib/systemd/systemd --user
pi         718  0.0  0.3  35304  3604 ?        S    00:19   0:00 (sd-pam)
pi         732  0.3  0.4  12240  4208 ?        S    00:19   0:06 sshd: pi@pts/0
pi         733  0.0  0.4   8492  3944 pts/0    Ss   00:19   0:01 -bash
root       833  0.0  0.6  12240  6248 ?        Ss   00:22   0:00 sshd: pi [priv]
pi         839  0.1  0.3  12240  3640 ?        S    00:23   0:02 sshd: pi@pts/1
pi         840  0.0  0.4   8492  3932 pts/1    Ss+  00:23   0:00 -bash
root      1024  0.0  0.0      0     0 ?        I    00:32   0:00 [kworker/1:0-
mm_percpu_wq]
root      1036  0.0  0.0      0     0 ?        I    00:33   0:00 [kworker/3:0-
events_power_efficient]
root      1175  0.0  0.0      0     0 ?        I<   00:41   0:00 [kworker/0:2H]
root       1194  0.0  0.0      0     0 ?        I    00:42   0:00 [kworker/2:1-
events]
root       1197  0.5  0.0      0     0 ?        I    00:42   0:03 [kworker/0:1-
events]
root       1227  0.0  0.0      0     0 ?        I    00:46   0:00 [kworker/0:3-
mm_percpu_wq]
root       1228  0.0  0.0      0     0 ?        I    00:46   0:00 [kworker/3:1-
events_freezable_power_]
root       1231  0.0  0.0      0     0 ?        I<   00:47   0:00 [kworker/2:0H]
root       1240  0.5  0.0      0     0 ?        I    00:47   0:02 [kworker/u8:0-
brcmf_wq/mmc1:0001:1]
root       1243  0.8  0.0      0     0 ?        I    00:47   0:03 [kworker/u8:3-
events_unbound]
root       1248  0.0  0.0      0     0 ?        I<   00:47   0:00 [kworker/3:0H]
root       1249  0.5  0.0      0     0 ?        I    00:47   0:02 [kworker/u8:4-
events_unbound]
root       1250  0.0  0.0      0     0 ?        I    00:47   0:00 [kworker/1:2-
mm_percpu_wq]
root       1251  0.0  0.0      0     0 ?        I<   00:47   0:00 [kworker/1:2H]
root       1297  0.0  0.0      0     0 ?        I<   00:50   0:00 [kworker/0:1H]
root       1310  0.0  0.0      0     0 ?        I    00:52   0:00 [kworker/0:0-
events]
root       1320  0.0  0.0      0     0 ?        I    00:53   0:00 [kworker/1:1-
events]
root       1326  0.0  0.0      0     0 ?        I    00:53   0:00 [kworker/u8:1]
root       1327  0.0  0.0      0     0 ?        I    00:53   0:00 [kworker/3:2]
root       1334  0.0  0.3  18592  3720 ?        S    00:54   0:00
/lib/systemd/systemd-udevd
root       1336  0.0  0.3  18592  3320 ?        S    00:54   0:00
/lib/systemd/systemd-udevd
pi         1338  0.0  0.2   9788  2460 pts/0    R+   00:54   0:00 ps aux
```

Appendix B – Exfiltrated Code and Text Files

HTML

HTML on Port 7245

```
<!DOCTYPE html>
<head>
   <title>Temperature Web Server</title>
   <!-- Latest compiled and minified CSS -->
   <link rel="stylesheet" href="/static/styles/bootstrap.min.css"
crossorigin="anonymous">
   <!-- Optional theme -->
   <link rel="stylesheet" href="/static/styles/bootstrap-theme.min.css"
crossorigin="anonymous">
</head>

<meta http-equiv="refresh" content="15">

<body>
    <h1>Temperature Web Server</h1>
    <br />

    <h2>Device (DEI)</h2>
    <form method="POST">
    <select id="SelectedDevice" name="SelectedDevice">

          <option value="70B3D5499C1AA8FB"
selected="selected">70B3D5499C1AA8FB</option>

    </select>
    <input type="submit" value="Select Device">
    </form>
    <br />
    <h2>Heating currently: OFF</h2>
    <h2>Room Currently: 29.0 &#8451</h2>
    <h2>Pressure: 1018.8mb </h2>
    <h2>Humidity: 36.5% </h2>
    <h2>Battery: 4.802V</h2>
    <h2>DevEUI: 70B3D5499C1AA8FB</h2>
    <h2>Name: Test LoPy4</h2>
</body>

</html>
```

HTML on Port 8245

```
<!DOCTYPE html>
<head>
```

```
   <title>PIR Web Server</title>
   <!-- Latest compiled and minified CSS -->
   <link rel="stylesheet" href="/static/styles/bootstrap.min.css"
crossorigin="anonymous">
   <!-- Optional theme -->
   <link rel="stylesheet" href="/static/styles/bootstrap-theme.min.css"
crossorigin="anonymous">
</head>

<meta http-equiv="refresh" content="15">

<body>
    <h1>PIR Web Server</h1>
    <br />

    <h2>Device (DEI)</h2>
    <form method="POST">
    <select id="SelectedDevice" name="SelectedDevice">

          <option value="E24F43FFFE44CD2A">E24F43FFFE44CD2A</option>

    </select>
    <input type="submit" value="Select Device">
    </form>
    <br />
    <h2>PIR Count: 0 activations </h2>
    <h2>Room Currently:  &#8451</h2>
    <h2>Battery:  %</h2>
    <h2>DevEUI: </h2>
  <h2>Name: </h2>

</body>

</html>
```

README.TXT

```
Hi Norman

Some notes on the temperature code for the PI

Cheers

Kenny

Install Mosquitto
sudo apt install -y mosquitto mosquitto-clients
sudo systemctl enable mosquitto.service

Install flask
sudo apt-get install python-pip python-flask
sudo pip install flask

Installing Python Paho-MQTT
sudo pip install -I flask-mqtt==1.0.1 – Use this version as we are still using
Python 2. Basically a wrapper over paho mqtt

Testing MQTT with MQTT-Explorer

Temperature Sever at: http://192.168.125.116:7245/
Debug Sever output at: http://192.168.125.116:7246/  -- Shows oldest event first
PIR Server at: http://192.168.125.116:8245/

Project files on PI directory: ~/hackathon/web-app
Creates debug log with MQTT messages and other control data in debug.log

Manual Launching

python temperature.py
python debug_output.py
python pir.py

MQTT Topics

Temperature
MQTT_TOPIC = "LRWAN_HEATING_CONTROL_UPLINK"
MQTT_HEATING_CONTROL_MODE = "LRWAN_HEATING_CONTROL_DOWNLINK/"

Decoder parser
fun(Fields, <<16#0073:16, Press:16, 16#0167:16, Temp:16, 16#0268:16,
Humidity,16#0300:16, Battery, 16#0401:16, LED>>) ->
  Fields#{ pressure => Press/10, temperature => Temp/10, humidity => Humidity/2,
battery => Battery, led => LED}
end.

PIR
MQTT_TOPIC = "LOPY_PIR_UPLINK"
MQTT_DOWNLINK = "LOPY_CONTROL_DOWNLINK/"

Decoder parser
fun(Fields, <<16#0000:16, PirCount:16, 16#0167:16, Temp:16,16#0201:16,
Battery:16>>) ->
  Fields#{pircount => PirCount, temperature => Temp/10, battery => Battery/1000}
```

```
end.

Install services
sudo systemctl enable xxxx.service

Have placed in /etc/systemd/system/ for now
```

TEMPERATURE.PY

```
#!/usr/bin/env python
#
# Simple temperature app
#

# Import  flask for webpage
from flask import Flask, flash, redirect, render_template, request, session, a
bort
from flask_mqtt import Mqtt
from datetime import datetime
import json
import os
import time
from sqlalchemy.orm import sessionmaker
from tabledef import *
engine = create_engine('sqlite:////var/local/temperature/temperature.db', echo
=False)

app = Flask(__name__)

# Temperature set value
off_temperature_value = 23

# Current heating mode
current_heat_mode = "OFF"
current_room_temperature = 23

# Lastest device message
message_data = {}

# Device list
device_list = set()

# Selected device
selected_device = "0"

# Session idle time last time used
session_last_time = datetime.now()

# Lock out counter
attempt_count = 0
```

```
# Define Variables for MQTT client
MQTT_TOPIC = "HEATING_CONTROL_UPLINK"
MQTT_HEATING_CONTROL_MODE = "HEATING_CONTROL_DOWNLINK/"

# Create MQTT
app = Flask(__name__)
app.config['MQTT_BROKER_URL'] = 'localhost'
app.config['MQTT_BROKER_PORT'] = 1883
app.config['MQTT_USERNAME'] = 'dev'
app.config['MQTT_PASSWORD'] = 'pa$$word'
app.config['MQTT_REFRESH_TIME'] = 1.0  # refresh time in seconds
mqtt = Mqtt(app)

####
## Processing
####

# Debug write
def write_to_log(text):
  time_stamp = str(datetime.now())
  line = '[' + time_stamp + '] - ' + text + '\n'

  filename = "/tmp/temperature_debug.log"

  if os.path.exists(filename):
    append_write = 'a+' # append if already exists
  else:
    append_write = 'w+' # make a new file if not

  with open(filename,append_write) as debugLog:
     debugLog.write(line)
     debugLog.close()

# Send heating off message
def send_heating_off():
  global off_temperature_value
  global current_heat_mode
  global selected_device

  # Publish back to server over MQTT

  # Message to send
  message = '{"data":"00"}'

  # Publish as JSON
  mqtt.publish(MQTT_HEATING_CONTROL_MODE + selected_device, message)
  current_heat_mode = 'OFF'
  # Debug message
  string_log = ('Sent message on topic {}: {}'.format(MQTT_HEATING_CONTROL_MOD
E, message))
  write_to_log(string_log);

# Send heating on message
def send_heating_on():
  global off_temperature_value
  global current_heat_mode
  global selected_device

  # Message to send
  message = '{"data":"01"}'

  # Publish as JSON
  mqtt.publish(MQTT_HEATING_CONTROL_MODE + selected_device, message)
  current_heat_mode = 'ON'

  # Debug message
  string_log = ('Sent message on topic {}: {}'.format(MQTT_HEATING_CONTROL_MOD
E, message))
  write_to_log(string_log);

# Check if heating should be turned off
def check_heating_mode(temperature):
  global off_temperature_value
  global current_heat_mode
  global current_room_temperature

  # Temperature above off level and is currently on
  if ( (temperature > off_temperature_value) and (current_heat_mode == 'ON') )
:
    send_heating_off()
  elif ( (temperature < off_temperature_value) and (current_heat_mode == 'OFF'
) ):
    send_heating_on()

  # Store the current room temperature
  current_room_temperature = temperature

####
## MQTT handlers
####

# Connect to MQTT callback
@mqtt.on_connect()
def handle_connect(client, userdata, flags, rc):
    mqtt.subscribe(MQTT_TOPIC)

# Call back for messages when received
@mqtt.on_message()
def handle_mqtt_message(client, userdata, message):
   global message_data
   global device_list
   global selected_device

   # Convert JSON to dict and store
   temp_message_data = json.loads(message.payload)

   # Add device needs added to set
   device_list.add(temp_message_data["deveui"])

   # If message from device selected
   if (selected_device == str(temp_message_data["deveui"])) :

     # Store message data as for device we care about
     message_data = temp_message_data

     print(json.dumps(message_data, indent=4, sort_keys=True))

     # Get the temperature value so can be processed
     current_temperature_value = int(message_data["temperature"])

     # Log message
     string_log = ('Received message on topic {}: {}'.format(message.topic, me
ssage.payload.decode()))
     write_to_log(string_log);

     # Check if heating mode needs updated
     check_heating_mode(current_temperature_value)

####
## Flask handlers
####

@app.context_processor
def inject_conf_in_all_templates():
    global message_data

    return dict(message_data=message_data)

# Create the main entry point
@app.route("/")
def main():
  global off_temperature_value
  global current_room_temperature
  global current_heat_mode
  global device_list
  global selected_device

  # Pass the template data into the template main.html and return it to the us
er
  return render_template("main_temperature.html", device_list = device_list, s
elected_device = selected_device, off_temperature_value = off_temperature_valu
e, current_heat_mode = current_heat_mode, current_room_temperature = current_r
oom_temperature)

@app.route('/', methods=['POST'])
def set_temp_form_post():
  global off_temperature_value
  global current_heat_mode
  global current_room_temperature
  global selected_device
  global message_data

  # Check for device in submitted data
  if "SelectedDevice" in request.form:
     # If changed then clear current data
     if selected_device != str(request.form['SelectedDevice']):
        message_data = {}
        selected_device = str(request.form['SelectedDevice'])

  return render_template("main_temperature.html", device_list = device_list, s
elected_device = selected_device, off_temperature_value = off_temperature_valu
e, current_heat_mode = current_heat_mode, current_room_temperature = current_r
oom_temperature)

# Add control page
@app.route("/control")
def control_temperature():
  # Check is valid session
  if not session.get('logged_in'):
    return render_template('login.html')
  else:
    return render_template("temperature_control.html", device_list = device_li
st, selected_device = selected_device, off_temperature_value = off_temperature
_value, current_heat_mode = current_heat_mode, current_room_temperature = curr
ent_room_temperature)

@app.route('/control', methods=['POST'])
def set_temp_control_form_post():
  global session_last_time
  # Calc idle period
  duration = datetime.now() - session_last_time

  # Check is valid session or idle timeout
  if not(session.get('logged_in')):
    return render_template("login.html")
  elif (duration.total_seconds() > 60):
    return render_template("login.html", message = "Idle timeout!")
  else:
    global off_temperature_value
    global current_heat_mode
    global current_room_temperature
    global selected_device
    global message_data

    # Reset idle timer
    session_last_time = datetime.now()

    # Check for device in submitted data
    if "SelectedDevice" in request.form:
       # If changed then clear current data
       if selected_device != str(request.form['SelectedDevice']):
          message_data = {}
          selected_device = str(request.form['SelectedDevice'])

    # Check if the submitted data has a new temperature
    if "Temperature" in request.form:
       off_temperature_value = int(request.form['Temperature'])

       # Check if heating mode needs updated as temperature changed
       check_heating_mode(current_room_temperature)

    # Check if the submitted data was the toggle control
    if "HeatingControlMode" in request.form:
        if request.form.get('HeatingControlMode') == 'Turn Heating Off':
          send_heating_off()
        elif  request.form.get('HeatingControlMode') == 'Turn Heating On':
          send_heating_on()

    return render_template("temperature_control.html", device_list = device_li
st, selected_device = selected_device, off_temperature_value = off_temperature
_value, current_heat_mode = current_heat_mode, current_room_temperature = curr
ent_room_temperature)

# Handles the login
@app.route('/login', methods=['POST'])
def do_admin_login():
  global session_last_time
  global attempt_count

  # If valid then redirect to the control page else return to home
  if ("username" in request.form) and ("password" in request.form):
    Session = sessionmaker(bind=engine)
    s = Session()
    POST_USERNAME = str(request.form['username'])
    POST_PASSWORD = str(request.form['password'])
    query = s.query(User).filter(User.username.in_([POST_USERNAME]), User.pass
word.in_([POST_PASSWORD]) )
    result = query.first()
    if result:
      # Reset attempts
      attempt_count = 0

      # Set session start
      session_last_time = datetime.now()
      session['logged_in'] = True
      return redirect("control", code=303)

    message = "Unknown User!"
    attempt_count = attempt_count + 1
    # Lock out
    if (attempt_count > 3):
      time.sleep(10)
      attempt_count = 0

    return render_template("login.html", message = message)

# Specifiy the port and addess for the webserver
if __name__ == "__main__":
   app.secret_key = os.urandom(12)

   DEBUG = False
   TESTING = False

   app.run(host='0.0.0.0', use_reloader=False, port=7245, debug=False)
```

DEBUG_OUTPUT.PY

```
#
# Simple temperature app
#

# Import  flask for webpage
from flask import Flask, render_template, request
```

```python
app = Flask(__name__)

####
## Flask handlers
####

# Create the main entry point
@app.route("/")
def main():

  filename = "/tmp/temperature_debug.log"

  with open(filename, "r") as f:
    content = f.read()

  # Pass the template data into the template main.html
  return render_template("debug.html", content=content)

# Specifiy the port and addess for the webserver
if __name__ == "__main__":
   app.run(host='0.0.0.0', use_reloader=False, port=7246, debug=False)
```

PIR.PY

```python
#!/usr/bin/env python
#
# Simple temperature app
#

# Import  flask for webpage
from flask import Flask, render_template, request
from flask_mqtt import Mqtt
from datetime import datetime
import json
import os

app = Flask(__name__)

# PIR count
pir_count = 0

# Current temp
current_temperature_value = 23

# Lastest device message
message_data = {}
```

```python
# Device list
device_list = set()

# Selected device
selected_device = "0"

# Define Variables for MQTT client
MQTT_TOPIC = "PIR_UPLINK"
MQTT_DOWNLINK = "PIR_DOWNLINK/"

# Create MQTT
app = Flask(__name__)
app.config['MQTT_BROKER_URL'] = 'localhost'
app.config['MQTT_BROKER_PORT'] = 1883
app.config['MQTT_USERNAME'] = 'dev'
app.config['MQTT_PASSWORD'] = 'pa$$word'
app.config['MQTT_REFRESH_TIME'] = 1.0  # refresh time in seconds
mqtt = Mqtt(app)

####
## Processing
####

# Debug write
def write_to_log(text):
  time_stamp = str(datetime.now())
  line = '[' + time_stamp + '] - ' + text + '\n'

  filename = "/tmp/pir_debug.log"

  if os.path.exists(filename):
    append_write = 'a+' # append if already exists
  else:
    append_write = 'w+' # make a new file if not

  with open(filename,append_write) as debugLog:
      debugLog.write(line)
      debugLog.close()

# Send heating off message
def send_control_message():
  global selected_device

  # Publish back to server over MQTT

  # Message to send
  message = '{"data":"00"}'
  # Publish as JSON
  mqtt.publish(MQTT_DOWNLINK + selected_device, message)
  current_heat_mode = 'OFF'

  # Debug message
  string_log = ('Sent message on topic {}: {}'.format(MQTT_DOWNLINK, message))
  write_to_log(string_log);

####
## MQTT handlers
####

# Connect to MQTT callback
@mqtt.on_connect()
def handle_connect(client, userdata, flags, rc):
    mqtt.subscribe(MQTT_TOPIC)

# Call back for messages when received
@mqtt.on_message()
def handle_mqtt_message(client, userdata, message):
   global message_data
   global device_list
   global selected_device
   global pir_count
   global current_temperature_value

   # Convert JSON to dict and store
   temp_message_data = json.loads(message.payload)

   # Add device needs added to set
   device_list.add(temp_message_data["deveui"])

   # If message from device selected
   if (selected_device == str(temp_message_data["deveui"])) :

     # Store message data as for device we care about
     message_data = temp_message_data

     print(json.dumps(message_data, indent=4, sort_keys=True))

     # Get the temperature value so can be processed
     current_temperature_value = int(message_data["temperature"])

     # Get PIR count
     pir_count = pir_count + int(message_data["pircount"])

     # Log message
     string_log = ('Received message on topic {}: {}'.format(message.topic, me
ssage.payload.decode()))
     write_to_log(string_log);

   # Update the page
   #return render_template("main_pir.html", device_list = device_list, selecte
d_device = selected_device, current_temperature_value = current_temperature_va
lue, pir_count = pir_count)

####
## Flask handlers
####

@app.context_processor
def inject_conf_in_all_templates():
    global message_data

    return dict(message_data=message_data)

# Create the main entry point
@app.route("/")
def main():
  global current_temperature_value
  global device_list
  global selected_device
  global pir_count

  # Pass the template data into the template main.html and return it to the us
er
  return render_template("main_pir.html", device_list = device_list, selected_
device = selected_device, pir_count = pir_count, current_temperature_value = c
urrent_temperature_value)

@app.route('/', methods=['POST'])
def set_temp_form_post():
  global current_temperature_value
  global selected_device
  global message_data
  global pir_count

  # Check for device in submitted data
  if "SelectedDevice" in request.form:
     # If changed then clear current data
     if selected_device != str(request.form['SelectedDevice']):
        message_data = {}
        pir_count = 0
        selected_device = str(request.form['SelectedDevice'])
  # Check if the submitted data has a new temperature
  if "Temperature" in request.form:
     current_room_temperature = int(request.form['Temperature'])

  # Check if the submitted data was the toggle control
  if "ResetCount" in request.form:
    pir_count = 0
  return render_template("main_pir.html", device_list = device_list, selected_
device = selected_device, current_temperature_value = current_temperature_valu
e, pir_count = pir_count)

# Specifiy the port and addess for the webserver
if __name__ == "__main__":
   app.run(host='0.0.0.0', use_reloader=False, port=8245, debug=False)
```

STYLE.CSS

```
* {
box-sizing: border-box;
}

*:focus {
outline: none;
}
body {
font-family: Arial;
background-color: #3498DB;
padding: 50px;
}
.login {
margin: 20px auto;
width: 300px;
}
.login-screen {
background-color: #FFF;
padding: 20px;
border-radius: 5px
}

.app-title {
text-align: center;
color: #777;
}

.login-form {
text-align: center;
```

```
}
.control-group {
margin-bottom: 10px;
}

input {
text-align: center;
background-color: #ECF0F1;
border: 2px solid transparent;
border-radius: 3px;
font-size: 16px;
font-weight: 200;
padding: 10px 0;
width: 250px;
transition: border .5s;
}

input:focus {
border: 2px solid #3498DB;
box-shadow: none;
}

.btn {
border: 2px solid transparent;
background: #3498DB;
color: #ffffff;
font-size: 16px;
line-height: 25px;
padding: 10px 0;
text-decoration: none;
text-shadow: none;
border-radius: 3px;
box-shadow: none;
transition: 0.25s;
display: block;
width: 250px;
margin: 0 auto;
}

.btn:hover {
background-color: #2980B9;
}

.login-link {
font-size: 12px;
color: #444;
display: block;
margin-top: 12px;
```

```
}
```