WEB VIEWABLE CAMERA

CMP408 – System Internals and Cyber Security

Jonah McElfatrick
1700463

+Contents

Introduction

This project aims to create a working system that allows a user to press a button, a camera will then take an image and upload it to AWS. The image will be displayed on a webpage to allow another user to be able to see the image. This is to act as a simple camera doorbell. This allows communication on a hardware level with the button, a software level with the capturing of the image, and a cloud level with the use of AWS to host the website and store the taken images.

Methodology

The methodology can be broken up into different sections, these include:

1) Wiring Setup
2) Modifying an LKM
3) C application that communicates with the LKM and will run a Python script on button press.
4) Python scripts that interacts with the Pi camera and takes a photo, uploads the photo to an AWS S3 bucket as well as the image filename to an RDS database.
5) Setting up the cloud. This includes setting up an AWS S3 bucket for storage of images. It also includes the initialising of an AWS EC2 instance to act as a web server and an RDS instance for storing filenames.
6) A web application for displaying of images retrieved from the S3 bucket.

1 – Wiring up the project

To setup the pi, a button was connected, one side to 3.3v and the other to GPIO pin 21. The pi camera cable was connected to the CSI camera connector using the corresponding cable. This can be seen in the image below.

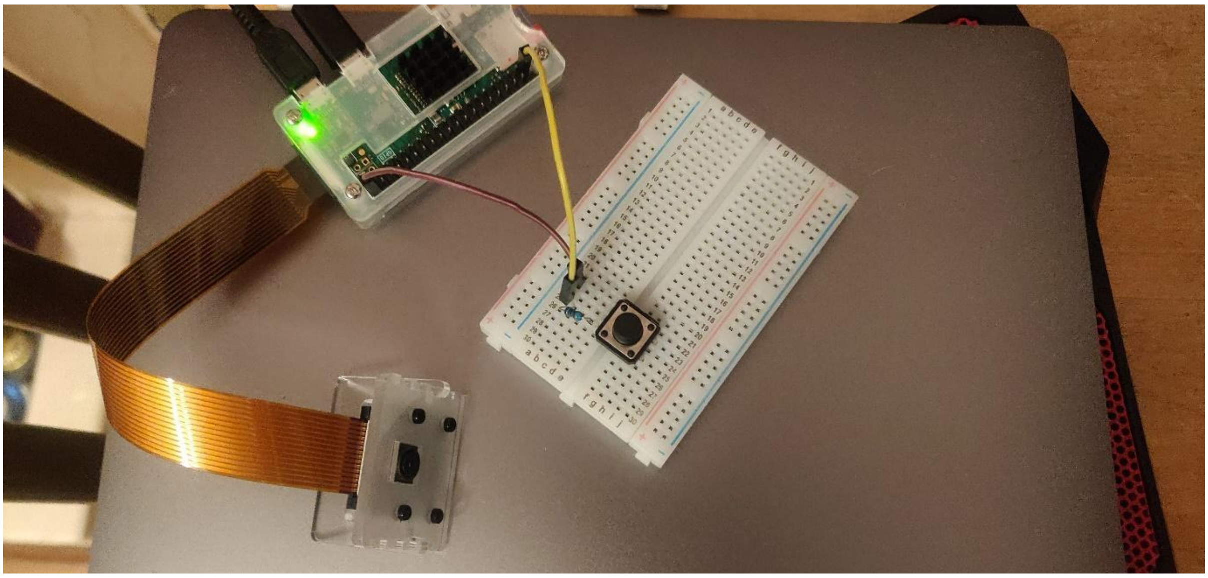


Figure 1: Wiring

To allow interaction between the pi and the camera, an interface option must be turned on. This can be completed by using the command 'raspi-config'. The screen in figure 2 below will be displayed.

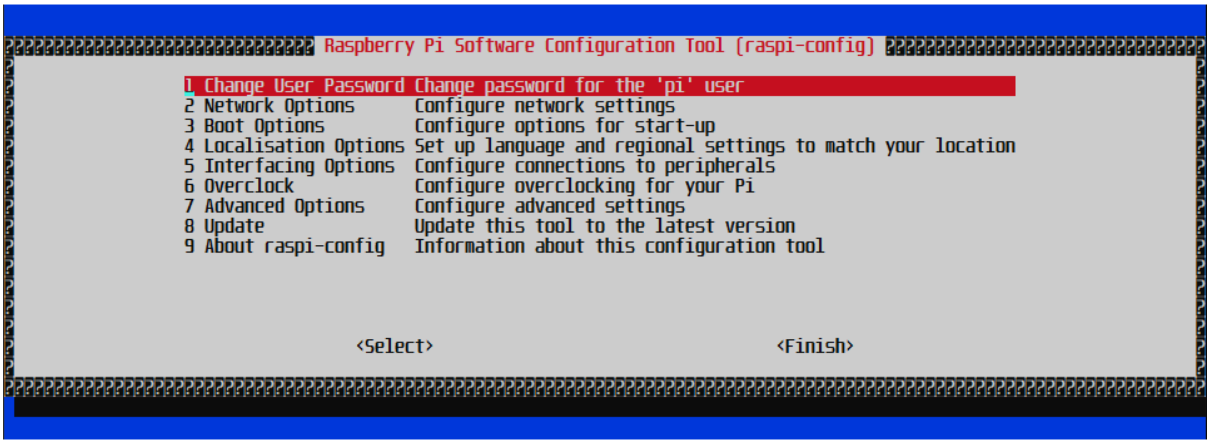


Figure 2:Config

From here, selecting 'Interfacing Options', 'P1 Camera' and 'Yes'. After a restart, the pi camera should be available to use.

2 – LKM and C application

Using the independent learning task 2 as a basis for this LKM, the purpose of the LKM is to read the status of the given GPIO pin and return the status of it to the user application to allow for the calling of a python script. This is accomplished through the following code.

```
case IOCTL_PIIO_GPIO_READ:
    memset(&apin , 0, sizeof(apin));
    copy_from_user(&apin, (gpio_pin *)arg, sizeof(gpio_pin));
    gpio_request(apin.pin, apin.desc);
    apin.value = gpio_get_value(apin.pin);
    strcpy(apin.desc, "LKMpin");
    copy_to_user((void *)arg, &apin, sizeof(gpio_pin));
    printk("piio: IOCTL_PIIO_GPIO_READ: pi:%u - val:%i - desc:%s\n" , apin.pin , apin.value , apin.desc);
    break;
```

Figure 3: LKM

To compile the LKM a fedora VM was used with the following command seen below.

```
[cmp408@localhost driver]$ make KERNEL=/home/cmp408/rpisrc/linux CROSS=/home/cmp408/tools/
arm-bcm2708/arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
```

Figure 4: Make File command

The user application will call the LKM repeatedly to check the status of the GPIO pin. When the status is returned as true, this will activate the python script. The main function can be seen in figure 5 below, where the LKM is called repeatedly to check the status of the GPIO pin passed in via a command line argument.

```c
int main(int argc, char *argv[]) {
    printf("User App\n");
    int fd, ret;
    char *msg = "Message passed by ioctl\n";

    fd = open("//dev//piiodev", O_RDWR);
    if (fd < 0) {
        printf("Can't open device file: %s\n", DEVICE_NAME);
        exit(-1);
    }

    if (argc > 1) {

        if (!strncmp(argv[1], "readpin", 8)) {
            /*  Pass GPIO struct with IO control */
            memset(&apin, 0, sizeof(apin));
            strcpy(apin.desc, "Details");
            apin.pin = strtol(argv[2], NULL, 10);
            /* Pass 'apin' struct to 'fd' with IO control*/
            while(1){
                ret = ioctl(fd, IOCTL_PIIO_GPIO_READ, &apin);
                printf("READ: pin:%i - val:%i - desc:%s\n", apin.pin, apin.value,
                apin.desc);
                if (apin.value == 1){
                    char * paramsList[] = { "/bin/bash", "-c",
                        "/usr/bin/python /home/pi/Desktop/cameraV2.py", NULL };
                    execv("/bin/bash",paramsList); // python system call
                }
            }
        }
    }

    close(fd);
    printf("Exit 0\n");

    return 0;
}
```

Figure 5:C Application

The full source code of the LKM can be found in appendix A and the C application in appendix B.

4 – Python Script

The script interacts with the camera and sets the resolution, rotation, and overlaying text parameters. This script saves the captured image locally and to the cloud. The function can be seen below.

```python
def captureImage():
        print ("\nCapturing Image...")
        #Get current date stamp for save file
        ts = datetime.datetime.now()
        #Initialise Camera
        camera = PiCamera()

        #Camera Settings
        camera.resolution = (1920,1080)
        camera.rotation = 180

        #Start capture process
        camera.start_preview(alpha=200)
        camera.annotate_background = Color("Blue")
        camera.annotate_foreground = Color("White")
        camera.annotate_text = "Front Door"
        camera.annotate_text_size = 50
        sleep(2)
        camera.capture('piImages/' + str(ts) + '.jpg')
        camera.stop_preview()
        print ("\nImage Captured!")
        uploadImage('piImages/' + str(ts) + '.jpg')
```

Figure 6: captureImage Function

This takes the created image file and uploads it to the specified S3 bucket. This is carried out using the boto3 library. The use of AWS CLI credentials is used for uploading the image to a S3 bucket. To store and access these credentials, a credentials file was used with boto3 sessions. To set this up, a folder was created in the 'home/pi' directory called '.aws'. In this directory a file called credentials was created with the following data inside:

```
[default]
aws_access_key_id=REPLACE_WITH_ACCESS_KEY_ID
aws_secret_access_key=REPLACE_WITH_SECRET_ACCESS_KEY
aws_default_region=us-east-1
aws_session_token=REPLACE_WITH_SESSION_TOKEN
```

Figure 7: .aws credentials File

With this file in place, the stored contents can be accessed using the boto3 library. The same library is used in order to be able to upload the image to the S3 bucket. The function uploadImage can be seen in figure 8 below.

```python
def uploadImage(filepath):
        print ("\nUploading file to S3 bucket...")
        file_dir, file_name = os.path.split(filepath)
        bucket_name = os.getenv('bucket')
        session = boto3.Session(profile_name='default')
        s3_resource = session.resource('s3')
        bucket = s3_resource.Bucket(bucket_name)
        bucket.upload_file(
                Filename=filepath,
                Key=file_name,
                #ExtraArgs={'ACL': 'public-read'}
        )
        print ("\nUpload Completed!")
        uploadDatatoRDS(file_name)
```

Figure 8: uploadImage Function

The filename of the image taken is uploaded to an RDS database. Different credentials must be used for connecting to the RDS, these credentials are stored as environment variables. These are stored in /etc/profile.

```
export host=REPLACE_WITH_HOST
export port=REPLACE_WITH_PORT
export user=REPLACE_WITH_USER
export passw=REPLACE_WITH_PASSWORD
export database=REPLACE_WITH_DATABASE_NAME
export bucket=REPLACE_WITH_BUCKET_NAME
```

Figure 9: Environment Variables

These variables can then be captured by the python script to communicate with the RDS instance. The function in which it does this can be seen below in figure 10.

```python
def uploadDatatoRDS(file_name):
        ts = datetime.datetime.now()
        print ("\nUploading filename to RDS...")
        host=os.getenv('host')
        port=int(os.getenv('port'))
        user=os.getenv('user')
        passw=os.getenv('passw')
        database=os.getenv('database')

        db = pymysql.connect(host, user=user,port=port,
                                passwd=passw, db=database)
        cursor = db.cursor()
        sql = '''insert into images(URL, date) values('%s', '%s')''' % (file_name, ts)
        cursor.execute(sql)
        db.commit()
        print ("\nUploaded Successfully!")
```

Figure 10: uploadDatatoRDS Function

Another python file called ‘userloop.py’ is used to repeatedly call the user application to be able to have the python script activate every time the button is pressed. The code can be found in appendix C.

5 - Cloud

Three aspects of AWS were used for cloud services. An EC2 instance to act as a web server, a S3 bucket to store the captured images and an RDS database to store the filenames of the images to recursively and dynamically display the images on the webpage. The EC2 and RDS were places inside the same VPC with communication to the RDS only available from withing this VPC and from the whitelisted IP’s.

6 – Web-interface

A simple web interface was created in order to display the taken images from the pi camera. This website includes a mixture of HTML, CSS, PHP and AJAX. The web interface consists of four pages, the login screen, home page, images page and about page. The images page is where

Security

In each of the different sections of this project, security has been taken into consideration and implemented where needs be. The implemented security features are described below:

LKM/C Application

The LKM was modified to remove any functions that are not used by the C application, in this case meant removing any write pin functions. This is to help prevent from any exploits being carried out by turning pins on or off from command line.

Python Script

Different credentials are used for uploading the image to the S3 bucket and uploading the details to the RDS database. The S3 credentials are stored in the ‘~/.aws/credentials’ file. The credentials used for the RDS database and the bucket name for uploading the image to the S3 bucket are stored in ‘/etc/profile’, this allows the credentials to be set on terminal start up. Both methods prevent any credentials to be viewed in the source code. Having the credentials in different locations also allow for a lower chance of both sets of credentials from being found and exploited if the machine is exploited.

Cloud

The RDS database and the EC2 instance have been placed in the same VPC with the inbound rules for the RDS set to only allow MYSQL/Aurora traffic from the EC2 instance and from the ip address of the raspberry pi. The EC2 instances inbound traffic rules have been set to only allow SSH traffic from the raspberry pi and to allow HTTP traffic from anywhere. This can be locked down to only allow the user to view the webpage from their home address if needs be. The S3 bucket ACL has been set so that only the bucket owner has permission to read and write to the bucket. Screenshots of the security settings for each of these cloud aspects can be found in appendix E.

It would have been preferred to have the images encrypted at rest and not public, however due to issues with the AWS PHP SDK and the raspberry pi, this was unable to be implemented, however bucket policies did allow for ip restricted access to the images, the policy can be seen in appendix E.

Web Interface

A login page was implemented into the web interface with sessions combined with one login to determine if the person attempting to view the images has the rights to. If the user has not logged into the login page and attempt to go to the images.php by manually changing the URL, they are redirected to the login page as the session is not set. The credentials are stored in an RDS database with security settings that can be found described in the Cloud section above, although another measure was taken to secure these credentials. The password stored in the database was hashed before being stored into the database. This is another security step that has been taken to help prevent from any sensitive information from being leaked when in transit or at rest. A logout button was implemented to terminate the session to increase security as to not allow anther user to press the 'back' button and view the webpage without logging in.

The credentials used for connecting to the RDS database are stored separately from the PHP file in the directory '/var/www/inc/dbinfo.inc', this file is then included in the PHP and therefore the credentials are not hardcoded into the code itself.

Conclusion

This project was successful as it covered the 3 main areas of the module, hardware, software and cloud. This was using an LKM in the communication between the user application and the button, a c application and a python script which allowed for an image to be taken using the camera and then uploaded to AWS. An EC2 instance, S3 bucket and RDS database were used in the cloud on AWS to store and host the website and images taken by the camera.

Word Count: 1500

References & Bibliography

10. API - picamera.camera Module — Picamera 1.10 documentation (no date). Available at: https://picamera.readthedocs.io/en/release-1.10/api_camera.html (Accessed: 23 December 2020).

C execute python script via a system call (no date) *Stack Overflow*. Available at: https://stackoverflow.com/questions/24973657/c-execute-python-script-via-a-system-call (Accessed: 23 December 2020).

Create an EC2 instance and install a web server - Amazon Relational Database Service (no date). Available at: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Tutorials.WebServerDB.CreateWebServer.html (Accessed: 19 December 2020).

Get an object Using the AWS SDK for PHP - Amazon Simple Storage Service (no date). Available at: https://docs.aws.amazon.com/AmazonS3/latest/dev/RetrieveObjSingleOpPHP.html (Accessed: 23 December 2020).

Getting started with the Camera Module - What next? | Raspberry Pi Projects (no date). Available at: https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/8 (Accessed: 23 December 2020).

'How to Call a C function in Python' (2018) *GeeksforGeeks*, 9 November. Available at: https://www.geeksforgeeks.org/how-to-call-a-c-function-in-python/ (Accessed: 23 December 2020).

How To Upload and Download Files in AWS S3 with Python and Boto3 | The Coding Interface (no date). Available at: https://thecodinginterface.com/blog/aws-s3-python-boto3/ (Accessed: 11 December 2020).

Okyol, O. (2020) *Using AWS RDS and Python together*, *Medium*. Available at: https://towardsdatascience.com/using-aws-rds-and-python-together-5718a6878e4c (Accessed: 23 December 2020).

salt - How to use PHP's password_hash to hash and verify passwords (no date) *Stack Overflow*. Available at: https://stackoverflow.com/questions/30279321/how-to-use-phps-password-hash-to-hash-and-verify-passwords (Accessed: 23 December 2020).

Appendix A – Driver

piDriver.h

```
#ifndef PIIO_H
#define PIIO_H

#include <linux/ioctl.h>

typedef struct lkm_data {
    unsigned char data[256];
    unsigned long len;
    char type;
} lkm_data;

typedef struct gpio_pin {
    char desc[16];
    unsigned int pin;
    int value;
    char opt;
} gpio_pin;

#define IOCTL_PIIO_READ 0x65
#define IOCTL_PIIO_GPIO_READ
0x67

#define  DEVICE_NAME "piiodev"
#define  CLASS_NAME  "piiocls"

#endif
```

PiDriver.c

```
// Modified LKM from ILT2 in CMP408 System Internals and Cyber Security
#include "piDriver.h"

#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/fs.h>
#include <asm/uaccess.h>
#include <linux/uaccess.h>

#include <linux/gpio.h>
#include <linux/cdev.h>
#include <linux/device.h>
#include <linux/seq_file.h>
```

```c
static int DevBusy = 0;
static int MajorNum = 100;
static struct class*  ClassName  = NULL;
static struct device* DeviceName = NULL;

lkm_data lkmdata;
gpio_pin gppin;

static int device_open(struct inode *inode, struct file *file){
    printk(KERN_INFO "piio: device_open(%p)\n", file);

    if (DevBusy)
        return -EBUSY;

    DevBusy++;
    try_module_get(THIS_MODULE);
    return 0;
}

static int device_release(struct inode *inode, struct file *file){
    printk(KERN_INFO "piio: device_release(%p)\n", file);
    DevBusy--;

    module_put(THIS_MODULE);
    return 0;
}

static int device_ioctl(struct file *file, unsigned int cmd, unsigned
long arg){
    int i;
    char *temp;
    char ch;

    printk("piio: Device IOCTL invoked : 0x%x - %u\n" , cmd , cmd);

    // Switch statement for cmd input from C application, limited to
read for security
    switch (cmd) {
    case IOCTL_PIIO_GPIO_READ:
        memset(&gppin , 0, sizeof(gppin));
        copy_from_user(&gppin, (gpio_pin *)arg, sizeof(gpio_pin));
        gpio_request(gppin.pin, gppin.desc);
        gppin.value = gpio_get_value(gppin.pin);
        strcpy(gppin.desc, "LKMpin");
        copy_to_user((void *)arg, &gppin, sizeof(gpio_pin));
        printk("piio: IOCTL_PIIO_GPIO_READ: pi:%u - val:%i - desc:%s\n"
, gppin.pin , gppin.value , gppin.desc);
        break;
```

```
    default:
            printk("piio: command format error\n");
    }

    return 0;
}

struct file_operations Fops = {
    .unlocked_ioctl = device_ioctl,
    .open = device_open,
    .release = device_release,
};

static int __init piio_init(void){
    int ret_val;
    ret_val = 0;

       printk(KERN_INFO "piio: Initializing the piio\n");
       MajorNum = register_chrdev(0, DEVICE_NAME, &Fops);
          if (MajorNum<0){
             printk(KERN_ALERT "piio: failed to register a major
number\n");
             return MajorNum;
          }
       printk(KERN_INFO "piio: registered with major number %d\n",
MajorNum);

       ClassName = class_create(THIS_MODULE, CLASS_NAME);
       if (IS_ERR(ClassName)){
          unregister_chrdev(MajorNum, DEVICE_NAME);
          printk(KERN_ALERT "piio: Failed to register device class\n");
          return PTR_ERR(ClassName);
       }
       printk(KERN_INFO "piio: device class registered\n");

       DeviceName = device_create(ClassName, NULL, MKDEV(MajorNum, 0),
NULL, DEVICE_NAME);
       if (IS_ERR(DeviceName)){
          class_destroy(ClassName);
          unregister_chrdev(MajorNum, DEVICE_NAME);
          printk(KERN_ALERT "piio: Failed to create the device\n");
          return PTR_ERR(DeviceName);
       }
       printk(KERN_INFO "piio: device class created\n");

    return 0;
}

static void __exit piio_exit(void){
       device_destroy(ClassName, MKDEV(MajorNum, 0));
```

```
        class_unregister(ClassName);
        class_destroy(ClassName);
        unregister_chrdev(MajorNum, DEVICE_NAME);
        gpio_free(gppin.pin);
        printk(KERN_INFO "piio: Module removed\n");
}
module_init(piio_init);
module_exit(piio_exit);
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Jonah McElfatrick");
MODULE_DESCRIPTION("Read GPIO Pin Driver");
MODULE_VERSION("0.2");
```

Makefile

```
KERNEL := /home/cmp408/rpisrc/linux
PWD := $(shell pwd)
obj-m += piDriver.o

all:
	make ARCH=arm CROSS_COMPILE=$(CROSS) -C $(KERNEL) M=$(PWD) modules
clean:
	make -C $(KERNEL) M=$(PWD) clean
```

Appendix B – User application

piUserapp.h

```
#ifndef PIIO_H
#define PIIO_H

#include <linux/ioctl.h>

typedef struct lkm_data {
    unsigned char data[256];
    unsigned long len;
    char type;
} lkm_data;

typedef struct gpio_pin {
    char desc[16];
    unsigned int pin;
    int value;
```

```
    char opt;
} gpio_pin;

#define IOCTL_PIIO_READ 0x65
#define IOCTL_PIIO_GPIO_READ
0x67

#define  DEVICE_NAME "piiodev"
#define  CLASS_NAME  "piiocls"

#endif
```

piUserapp.c

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/ioctl.h>

#include"piUserapp.h"

gpio_pin gppin;
lkm_data lkmdata;


int main(int argc, char *argv[]) {
    printf("User App\n");
    int pidevopen, lkmreturn;
    char *msg = "Message passed by ioctl\n";

    pidevopen = open("//dev//piiodev", O_RDWR);
    if (pidevopen < 0) {
        printf("Can't open device file: %s\n", DEVICE_NAME);
        exit(-1);
    }

    // Checks to see if any additional arguments have been passed in
through the cli
    if (argc > 1) {

        // Checks to see if the command entered is the "readpin" command
        if (!strncmp(argv[1], "readpin", 8)) {
            /*  Pass GPIO struct with IO control */
            memset(&gppin, 0, sizeof(gppin));
            strcpy(gppin.desc, "Details");
```

```
            gppin.pin = strtol(argv[2], NULL, 10);
            /* Pass 'gppin' struct to 'pidevopen' with IO control*/

            // While loop to repeatedly call the lkm untill a button
press is registered
            while(1){
                // Calling the lkm
                lkmreturn = ioctl(pidevopen, IOCTL_PIIO_GPIO_READ,
&gppin);
                // Printing out the value of the pin
                printf("READ: pin:%i - val:%i - desc:%s\n", gppin.pin,
gppin.value,
                gppin.desc);
                // Checks to see if the button has been pressed
                if (gppin.value == 1){
                    // Calls the python script cameraV2.py if button has
been pressed
                    char * paramsList[] = { "/bin/bash", "-c",
                        "/usr/bin/python /home/pi/Desktop/cameraV2.py",
NULL };
                    execv("/bin/bash",paramsList);
                }
            }
        }
    }

    close(pidevopen);
    printf("Exit 0\n");

    return 0;
}
```

Makefile

```
#make
/home/cmp408/tools/arm-bcm2708/arm-linux-gnueabihf/bin/arm-linux-gnueabi
hf-gcc piUserapp.c
all: piUserapp.c

/home/cmp408/tools/arm-bcm2708/arm-linux-gnueabihf/bin/arm-linux-gnueabi
hf-gcc -g -Wall -o piUserapp piUserapp.c

clean:
      $(RM) piUserapp
```

Appendix C – Python Files

cameraV2.py

```
from picamera import PiCamera, Color
from time import sleep
import datetime, os, fcntl, boto3, pymysql

def main():
        captureImage()

def captureImage():
        print ("\nCapturing Image...")
        #Get current date stamp for save file
        ts = datetime.datetime.now()
        #Initialise Camera
        camera = PiCamera()

        #Camera Settings
        camera.resolution = (1920,1080)
        camera.rotation = 180

        #Start capture process
        camera.start_preview(alpha=200)
        camera.annotate_background = Color("Blue")
        camera.annotate_foreground = Color("White")
        camera.annotate_text = "Front Door"
        camera.annotate_text_size = 50
        sleep(2)
        camera.capture('piImages/' + str(ts) + '.jpg')
        camera.stop_preview()
        print ("\nImage Captured!")
        uploadImage('piImages/' + str(ts) + '.jpg')

def uploadImage(filepath):
        print ("\nUploading file to S3 bucket...")
        file_dir, file_name = os.path.split(filepath)
        bucket_name = os.getenv('bucket')
        session = boto3.Session(profile_name='default')
        s3_resource = session.resource('s3')
        bucket = s3_resource.Bucket(bucket_name)
        bucket.upload_file(
                Filename=filepath,
                Key=file_name,
                #ExtraArgs={'ACL': 'public-read'}
        )
        print ("\nUpload Completed!")
        uploadDatatoRDS(file_name)

def uploadDatatoRDS(file_name):
```

```
        ts = datetime.datetime.now()
        print ("\nUploading filename to RDS...")
        host=os.getenv('host')
        port=int(os.getenv('port'))
        user=os.getenv('user')
        passw=os.getenv('passw')
        database=os.getenv('database')

        db = pymysql.connect(host, user=user,port=port,
                                   passwd=passw, db=database)
        cursor = db.cursor()
        sql = '''insert into images(URL, date) values('%s', '%s')''' %
(file_name, ts)
        cursor.execute(sql)
        db.commit()
        print ("\nUploaded Successfully!")

main()
```

userloop.py

```
import os

while (1):
        os.system("./piUserapp readpin
21")
```

Appendix D –Web Interface

logout_handler.php

```
<?php
      session_start();
      unset($_SESSION['username']);
      session_destroy();
      session_write_close();
      header('Location:
..\loginPage.php');
?>
```

login_handler.php

```
<!--Created By Jonah McElfatrick -->
<?php
      ini_set('display_errors', 1);
      ini_set('display_startup_errors',1);
      error_reporting(E_ALL);
```

```
	session_start();
	require('connection.php');

	$username = test_input($_POST['username']);
	$password = test_input($_POST['password']);

    //Select logins from database and check against entered
	$sql = "select * from logins";
	$result = $conn->query($sql);
	if ($result->num_rows > 0){
		while($row = $result->fetch_assoc()) {
			if (password_verify($password,$row["userPassword"])) {
				$_SESSION['username'] = $username;
				header('Location:..\home.php');
			}
			else{
				echo "<p>You have entered the wrong 
credentials</p>";
				echo "<p>Please try again</p>";
				header('Refresh: 2; URL = ..\loginPage.php');
			}
		}
	}
	mysqli_close($conn);

	//Checks to see if the inputted data is in a valid format
	function test_input($data){
		$data = trim($data);
		$data = stripslashes($data);
		$data = htmlspecialchars($data);
		return $data;
	}
?>
```

connection.php

```
<?php
    include "/var/www/inc/dbinfo.inc";
    $host = DB_SERVER;
    $user = DB_USERNAME;
    $pass = DB_PASSWORD;
    $database = DB_DATABASE;
    $connection;

    $conn = mysqli_connect($host, $user, $pass, $database) or 
die("Connection failed: ". mysqli_connect_error());

    // Check connection
    if ($conn->connect_error){
        echo("Connection failed: " . mysqli_connect_error());
```

```
        exit();
    }
?>
```

about.php

```
<?php
    session_start();
    if(!isset($_SESSION['username']))
    {
        header('Location: loginPage.php');
    }

?>
<!DOCTYPE html>
<html lang="en" dir="ltr">
    <head>
        <meta charset="utf-8">
        <link rel= "stylesheet" type= "text/css"
href='static/styles.css' />
        <title>About Project</title>
    </head>
    <header>
                <div class="container">
                    <h1 class="logo"></h1>
                    <strong><nav>
                        <ul class="menu">
                            <li><a href="home.php">Home</a> </li>
                            <li><a href="images.php">Images</a></li>
                            <li><a href="about.php">About</a></li>
                            <li><a href="PHP/logout_handler.php">Log
Out</a></li>
                        </ul>
                    </nav></strong>
                </div>
            </header>
    <body>
        <div class="column">
            <h1> About this project </h1>
            <p> This project has the aim to be able to bring the
functionality of products such as the famous Ring Doorbell to those of a
tighter budget, or those who want to be able to be in control of their
own devices and security.</p>
        </div>
        <div class="column">
            <h1> Specifications </h1>
            <p>The hardware used in this project can be seen listed
below:</p>
```

```
            <li>Raspberry Pi Zero W</li>
            <li>Raspberry Pi Camera V2</li>
            <li>4 pin push button</li>
            <li>Resistor</li>
            <li>Female to male wires</li>
        </div>
        <div class="column">
            <h1> Software </h1>
            <p> This project uses different types of software and 
connections to bring all the functionality together. A list below is 
what software is used and follows is a description of how it is used: 
</p>
            <li>Raspbian</li>
            <li>LKM (Linux Kernel Module) in C</li>
            <li>C User Application</li>
            <li>Python</li>
            <li>Website</li>
            <ul>
                <li>HTML/CSS</li>
                <li>PHP/AJAX</li>
            </ul>
            <li>AWS (Amazon Web Service)</li>
            <ul>
                <li>EC2</li>
                <li>RDS</li>
                <li>S3</li>
            </ul>
        </div>
    </body>
</html>
```

home.php

```
<?php
    session_start();
    if(!isset($_SESSION['username']))
    {
        header('Location: loginPage.php');
    }
```

```
?>
<!DOCTYPE html>
<html lang="en" dir="ltr">
    <head>
        <meta charset="utf-8">
        <link rel= "stylesheet" type= "text/css"
href='static/styles.css' />
        <title>Project Home Page</title>
    </head>
    <header>
        <div class="container">
            <h1 class="logo"></h1>
            <strong><nav>
                <ul class="menu">
                    <li><a href="home.php">Home</a> </li>
                    <li><a href="images.php">Images</a></li>
                    <li><a href="about.php">About</a></li>
                    <li><a href="PHP/logout_handler.php">Log
Out</a></li>
                 </ul>
            </nav></strong>
        </div>
    </header>
    <body>
        <div class="description">
            <h1>Overview</h1>
            <p> Breaking down the main steps in this project can be seen
as follows: </p>
            <li>A LKM (Linux Kernel Module) running in the background
waiting to be called</li>
            <li>A C user application calling the LKM repeatedly to check
the status of the GPIO pin</li>
            <li>User presses the button and the status of the GPIO pin
is returned to the C application as on</li>
            <li>User application runs a python script</li>
            <li>The python script takes an image using the camera module
and uploads the image to AWS S3 and the filename to AWS RDS</li>
            <li>This website, hosted on an AWS EC2 instance, pulls the
filenames from the RDS database and pulls the file from the S3 bucket to
then be displayed on the screen</li>
        </div>
    </body>
</html>
```

images.php

```
<?php
    session_start();
```

```
    if(!isset($_SESSION['username']))
    {
        header('Location: loginPage.php');
    }

?>
<!DOCTYPE html>
<html lang="en" dir="ltr">
    <head>
        <meta charset="utf-8">
        <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.3/jquery.min.js"></sc
ript>
        <script language="javascript" type="text/javascript"></script>
        <link rel= "stylesheet" type= "text/css"
href='static/styles.css' />
        <title>Images Page</title>
    </head>
    <body>
        <div id="images">
            <header>
                <div class="container">
                    <h1 class="logo"></h1>
                    <strong><nav>
                        <ul class="menu">
                            <li><a href="home.php">Home</a> </li>
                            <li><a href="images.php">Images</a></li>
                            <li><a href="about.php">About</a></li>
                            <li><a href="PHP/logout_handler.php">Log
Out</a></li>
                        </ul>
                    </nav></strong>
                </div>
            </header>
            <h1>Captured Image</h1>
            <p>Here are the images captured from your Raspberry pi
camera</p>
            <?php
                ini_set('display_errors', 1);
                ini_set('display_startup_errors', 1);
                error_reporting(E_ALL ^ E_NOTICE);

                require("PHP/connection.php");

                $sql = "select * from images";

                $result = $conn->query($sql);
                if ($result->num_rows > 0){
                        while($row = $result->fetch_assoc()) {
                        echo '<h3>'.$row["URL"].'</h3>';
```

```
                        echo '<p><img
src="https://cmp408classtest.s3.amazonaws.com/'.$row["URL"].'"/></p>';
                        }
                }
                else{
                        echo "No images to show!";
                }
            ?>
        </div>
    </body>
    <script>
        function loadImages(){
            $('#images').load('images.php',function () {
                $(this).unwrap();
            });
        }
        loadImages(); // This will run on page load
        setInterval(function(){
            loadImages() // this will run after every 5 seconds
        }, 5000);
    </script>
</html>
```

loginPage.php

```
<!--Created By Jonah McElfatrick -->
<!DOCTYPE html>
<html lang="en">
<head>
      <title>Log-in Page</title>
      <meta name="viewport" content="width=device-width,
initial-scale=1">
      <meta charset="utf-8">

      <!-- Links the css stylesheet -->
    <link rel= "stylesheet" type= "text/css" href='static/styles.css' />
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.
css">

      <!--Linking javascript scripts-->
      <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"><
/script>
      <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js
"></script>
      <script type="text/javascript"
src="JavaScript/jscript.js"></script>
</head>
```

```
<body>
      <!-- Modal for loggin in the user-->
      <div class="modal-dialog">
            <div class="modal-content">
                  <div class="modal-header">
                        <h4><span class="glyphicon
glyphicon-lock"></span> Login In</h4>
                  </div>
                  <div class="modal-body">
                        <form role="form" action="PHP/login_handler.php"
method="post">
                              <div class="form-group">
                                    <label for="username"><span
class="glyphicon glyphicon-user"></span> Username</label>
                                    <input type="username"
name="username" class="form-control" id="username" required
placeholder="Enter Username?">
                              </div>
                              <div class="form-group">
                                    <label for="password"><span
class="glyphicon glyphicon-password"></span> Password</label>
                                    <input type="password"
name="password" class="form-control" id="password" required
placeholder="Enter Password">
                              </div>
                              <button type="submit" class="btn
btn-block">Login
                                    <span class="glyphicon
glyphicon-ok"></span>
                              </button>
                        </form>
                  </div>
            </div>
      </div>

</body>
</html>
```

styles.css

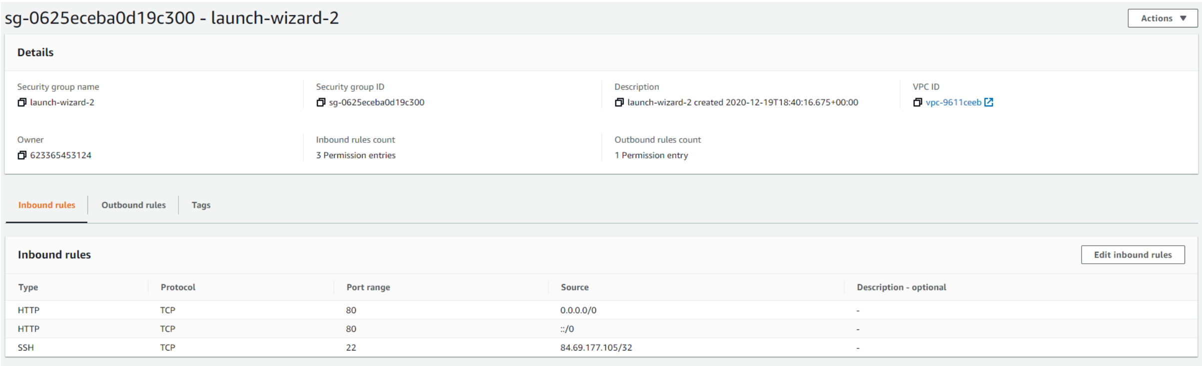
```css
* {
  box-sizing: border-box;
}

body {
    font-family: Arial, Helvetica, sans-serif;
    background: linear-gradient(90deg, rgba(255,255,255,1) 0%,
rgba(181,181,190,1) 0%, rgba(71,143,235,1) 100%);
}

/* Style the header */
.header {
    background-color: #f1f1f1;
    padding: 30px;
    text-align: center;
    font-size: 35px;
    color: White;

}

/* Styling the logo */
.logo{
    width:100%;
    height:110px;
    margin:0px;
    padding-top: 10px;
    padding-bottom: 10px;
    background-image: url("../images/banner.jpg");
    background-repeat: no-repeat, repeat;
    background-size: cover;
    background-position: center;
}

/*Start of navigation menu styling */
.menu{
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    background-color: #333;
}

.menu li{
    float:left;
}

.menu li a {
    display: block;
```

```
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}

.menu li a:hover {
  background-color: #111;
}
/* End of navigation menu styling */

/* Start of the styling of the columns */
.column {
    float: left;
    width: 32%;
    height: 400px;
    margin-left: 8px;
    margin-right: 9px;
    margin-right: 9px;
    margin-top: 20px;
    padding: 5px;
    text-align: justify;
    outline-color: dimgrey;
    outline-width: 8px;
    outline-style: inset;
}

.column h1{
    text-align: center;
}

.column ul{
    text-align: justify;
    margin-top: 3px;
    margin-bottom: 3px;
}

.column li{
    text-align: justify;
    margin-top: 3px;
    margin-bottom: 3px;
}
/* End of the styling of the columns */

img {
    float: left;
    width: 24%;
    height: 400px;
    margin:10px;
    margin-left: 8px;
```

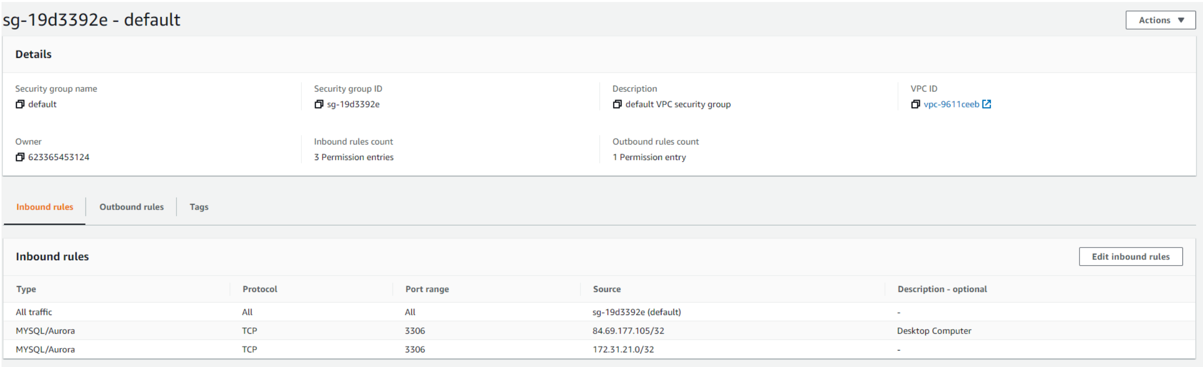
```
    margin-right: 9px;
    margin-right: 9px;
    margin-top: 20px;
    padding: 5px;
    text-align: justify;
}

h3{
    font-size: 14px;
    float: none;
}

/* Start of the description styling on home page */
.description{
    text-align: justify;
    float: inherit;
    margin: 10px;
}

.description li {
    text-align: justify;
}

/* Clear floats after the columns */
.row:after {
    content: "";
    display: table;
    clear: both;
}

/* Style the footer */
.footer {
    background-color: #f1f1f1;
    padding: 10px;
    text-align: center;
}

/* Responsive layout - makes the three columns stack on top of each
other instead of next to each other */
@media (max-width: 600px) {
  .column {
    width: 100%;
  }
}
```

Appendix E – Cloud Security Settings

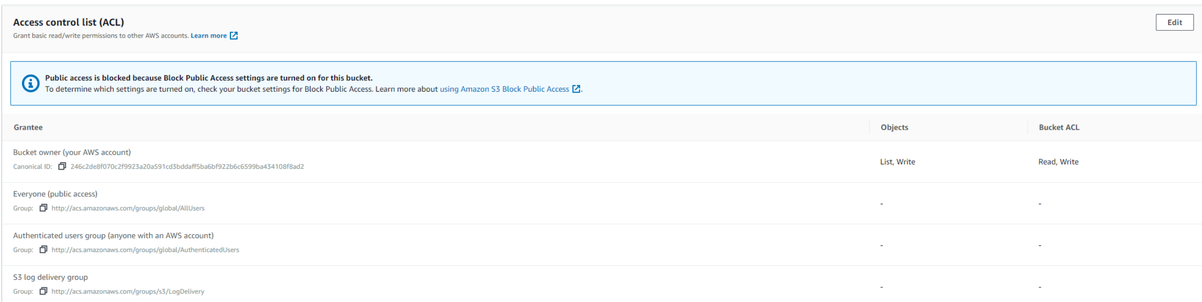
EC2



RDS



S3



Bucket Policy

```
{
  "Version": "2012-10-17",
  "Id": "S3PolicyId1",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::cmp408classtest/*",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": [
            "84.69.177.105",
            "172.31.21.0"
          ]
        }
      }
    }
  ]
```

VPC

Inbound rules | Outbound rules | Tags

Inbound rules Edit inbound rules

Type	Protocol	Port range	Source	Description - optional
All traffic	All	All	sg-19d3392e (default)	-
MYSQL/Aurora	TCP	3306	84.69.177.105/32	Desktop Computer
MYSQL/Aurora	TCP	3306	84.69.177.10/32	Raspberry Pi

Appendix F – Database Structure

Column Name	#	Data Type	Not Null	Auto Increment	Key
username	1	varchar(20)	☑	☐	PRI
userPassword	2	varchar(70)	☑	☐	

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
id	1	int	☑	☑	PRI		auto_increment
URL	2	text	☑	☐			
date	3	datetime	☑	☐			