Boston University
Electrical and Computer Engineering
EC464 Senior Design Project

2nd Prototype Test Plan

Team 24
Community Security

Samuel Evans evanss@bu.edu
Esteban Hernandez hesteban@bu.edu
Daniel-John Morel djmorel@bu.edu
Steve Numata numatas8@bu.edu
Cameron MacDonald Surman csms@bu.edu

February 25th, 2020

1.0 Introduction

For our second prototype testing, work revolved entirely around our data stream and the connections that needed to be made between the camera, AWS, and the user website. These connections are fundamental to the overall system for they allow us to report on events, conduct analysis, and then provide that information to those within the community.

We broke our testing up into the different points the data would pass through. Our first test is focused on recording motion-based events, syncing with S3, and then storing those clips on AWS. While seemingly simplistic, there are multiple moving parts that needed to be addressed to make this data transfer organized and efficient. The second test addressed the next connection, AWS to the user website. This involves footage uploaded into S3 to trigger an AWS Lambda handling function to insert it into a database hosted by AWS RDS, and ensuring it can be extracted by the user interface within the web application. Finally, we tested the eventual implementation of license plate recognition software through test scripts to ensure that the tools we plan on using will function as we need them to.

2.0 Raspberry Pi & AWS S3 Data Connection

Connecting the Raspberry pi's videos to AWS S3 storage requires multiple components to work together. The three main components are the motionEye program to collect videos, the AWS S3 library to sync videos, and the software utility cron to schedule the sync command. All of these components have been installed on the pi, and are configured to work with each other. This test will demonstrate that the pi can record motion-detected videos and routinely send them to S3 storage automatically.

2.1 Measurable Criteria

Since this test looks at uploading videos to S3 storage, the measurable criteria is more binary by nature. By this we mean that a video recorded by the Raspberry pi can be sent to S3 storage and then viewed from there. As such, the results will be determined either as a pass or failure based on if the recorded video is accessible in the AWS S3 console.

2.2 Testing Procedure

Most of the testing procedure is automatically done behind the scenes on the pi as the motionEye records motion-detected clips and cron transfers the videos to S3 storage via the AWS sync command. Since the pi is configured to automatically send videos at a given interval (cron is configured to sync every 2 minutes), much of the testing procedure is done by observation.

During the test, we will have the camera record the audience for approximately 10 seconds or so. While this is being done, we will show the audience that the pi is recording video by viewing its camera directory through the terminal and through the motionEye HTTP interface as seen in previous demos. Unlike previous demos, we will then show that the videos arrive to S3 storage via the AWS console. Upon a successful sync, we will then play back the video from the AWS S3 console to prove that it is viewable and can be later used in the project pipeline. Note that this entire testing procedure will be done using the pi's cellular shield to prove that such an internet connection works for the camera system.

2.3 Test Deliverables

The test will demonstrate that a motion-detected video can be captured by the pi, and then synced to our AWS S3 storage automatically. A successful test will show the captured video in S3 storage, which can then be viewed via the linked object URL.

3.0.1 Pulling Video From AWS to Display to User

3.1 Measurable Criteria

The measurable criteria for this test is in two parts. The first measurable criteria is that the database hosted on AWS RDS matches the information that is displayed in the Django admin page. This signifies that the Django framework is properly recognizing the information stored in the Postgresql database. The second part of the test is a pass fail test that determines whether the proper video footage is loaded into the web application's iframe given a camera, date, and time combination.

3.2 Testing Procedure

The testing procedure for ensuring that the web application is properly communicating to the AWS RDS database is to navigate to the Django administration page to make insertions into the database. The test is verified if the data inserted into the administration page is reflected in the Postgresql database, which can be viewed and queried using pgAdmin4. The opposite can also be verified, in that inserting an entry in pgAdmin4 will be reflected on the database. The second part of the testing procedure is to ensure that the proper footage is being loaded into the iframe. This can be done by looking at a footage entry in the database, looking at its corresponding date and camera relation, selecting the same camera and date in the web application, and confirming that the same footage entry selected from the database is being viewed.

3.3 Test Deliverables

The test will demonstrate that data pulled from the Postgresql database can be hosted by AWS RDS, and that data from it can be successfully retrieved from it, and the video can be displayed on the user interface. The test also demonstrates that the user selecting a combination of camera id, date, and the time the footage was taken will display the correct footage.

3.0.2 Automatically Sending Data from S3 to Postgresql Database

3.1 Measurable Criteria

The measurable criteria for this test is confirmation that the RDS Database has been uploaded properly according to data sent to it from AWS Lambda. In addition, assuming the functionality of the database to web application communication is valid, and a proper S3 trigger is set up, when the Raspberry pi camera module uploads a video to S3, Lambda should automatically update the database with an entry containing the object file's URL.

3.2 Testing Procedure

The testing procedure is split into an isolated test case and an overall system test case. The isolated test case consists of running an AWS Lambda test case containing the URL to an existing object file in S3, running it, and confirming it has been added to the "polls_footage_time" table in the Postgresql database. The second test is to have the Raspberry pi camera module upload a video file, and use pgAdmin4 to confirm that the video URL has been entered into "polls_footage_time" table in the database. Consequently, if Test 3.0.1 passes, it should be viewable on the web application as well.

3.3 Test Deliverables

The test will demonstrate that when a video footage is uploaded to S3 in the form of an mp4 video, AWS lambda will automatically add the object file's URL to the Postgresql database hosted by RDS. The test does not yet account for automatically sorting the footage into the correct date, as it will have to be done manually.

4.0 Testing OpenALPR Scripts on Test Clips

4.1 Measurable Criteria

The measurable criteria for this test will be to have a command that uploads license plate images or video feed. Once the files are uploaded we can then verify that the license plates were processed by checking the OpenALPR cloud.

4.2 Testing Procedure

The testing procedure will take about 20 license plate images and upload them to OpenALPR through a Python program. The Python program will call on the OpenALPR API that will then upload the images to the OpenALPR cloud to process the images. Once the images are processed, we can check the valuable information on OpenALPR. The information is then returned by the API through a JSON data file.

4.3 Test Deliverables

The deliverables will show up on OpenALPR and that will be the car’s license plate number, model, type, color, ect.