GONZAGA UNIVERSITY

School of Engineering and Applied Science
Center for Engineering Design and Entrepreneurship

Status Report
December 14 2017

Credential Security API with Facial Recognition

Prepared by:

[illegible]

Anton Sebastian Vargas

[illegible]

Elijah Michaelson

[illegible]

Brian Mackessy

1 Project Overview

1.1 Project Summary

Over the past decade, many major companies of have been hit with massive hacks that have resulted in hundreds of millions of people's private information to be stolen. The old method of username and password are far too susceptible to hacks, and just simply aren't good enough to keep people's private data secure. Vulnerabilities, such as the reuse of passwords or the use of weak passwords, are regularly exploited by malicious agents and it is hard to enforce policies on users that protect them. Another problem with passwords is that if you are smart and have many different passwords, it can be almost impossible to keep track of all of them. Because of these problems with usernames and passwords we think that the only way forward is biometric security. What we are proposing to use your face as your password. Advances in machine learning have made it possible for computers to recognize faces with a high degree of accuracy, opening the door for a secure and easy to use credential policy system.

We plan to build an API for software developers to integrate into their login pages to allow their users to save their usernames and passwords in our service, and automatically access these usernames and passwords when they pass a facial recognition test. Our software will be easy to use and free, with the goal of attracting as many everyday users as possible. We also want to make sure that our software can work on any computer that has a camera on it, something that existing software doesn't do.

1.2 Project Deliverables

Our primary deliverable for this project is an API that a developer could plug into their website to add facial recognition access to their website. The API needs to be simple enough for a programmer to use. The interface with the actual users needs to be simple enough from someone who doesn't know how to program to use. We want the users to be able to manage passwords though the already existing Google Credential software. Another deliverable is that the API needs to have minimal false positives in order to protect the user's information. The program also needs to automatically input the credentials when the facial recognition has been passed.

2 Work Accomplished

This past semester, we have made good headway on our project as we are right on schedule. The main work we have done:

2.1 Creating a proof of concept webpage

We created a Flask-server based website that has a simple login and register feature which we used to showcase our API

2.2 Accessing the camera and capture images from the web page to a python script

This is a user-story that we thought would be easy but ended up taking up a long time since it was broken up into accessing the camera, then taking the picture, then having that sent to the classifier python script

2.3 Face data classification

We have a face classification python script written that can be trained by the camera image can return a boolean to say if that image has been registered, as well as map the face to a set of credentials

2.4 A small API that provides minimal function calls for web developers

We currently have a python API for web developers with a Flask backend, but will scale/modify our API to accommodate other popular backends/frameworks such as php and react.js

2.5 A controlled, deployable virtual environment (container) for our work through Docker

We have a deployable Docker container that runs our software that can be scaled regardless of system due to the fact that the container holds all of the dependencies needed to run our software.

3 Base Schedule and Deviations

Our plan for this semester was to implement an API that could be plugged into websites to perform facial recognition. With the project demo in mind, our expectation for the API is that we could use function calls in fake website we created. This is very different from creating an API that **anyone** could plug into **their** website, which is the main goal of the project. Secondly, we wanted to have a classifier that could recognize faces better than random guessing. It was not a necessity that the classifier would always work, just that there was proof it was in the early stages of recognizing faces.

So far, we have accomplished both of those goals and have even exceeded our own expectations. It is important to note that new problems which have arisen have cancelled out any extra work we have done this semester, meaning that we are neither behind nor ahead of schedule. Our API is very simple, and works well for the website we build through Flask. The classifier is excellent at recognizing faces, but can easily be fooled by attempts to trick it such as putting a photo of a person in front of the camera. Other work we have accomplished this semester is creating a Docker container for our project to make it scalable. This has ended up being necessary because most of the program we have written so far are custom build for Unix systems.

4 Revised Schedule and Scope

When we first created our plan for the second semester we really had no accurate gauge for what we would accomplish this semester and what new problems would arise. Since we got ahead on a lot of work, while new problems also arose, we had to entirely scrap our plan for the remainder of December and the Spring semester.

2nd Week in Dec to 3rd Week in Dec:

Unit Testing, Regression Testing, and Accuracy Testing for all existing code

4th Week in Dec to 1st Week in Jan:

Modify classifier to detect attempts to fool software

2nd Week in Jan to 3rd Week in Jan:
Retesting classifier after modification for accuracy

4th Week in Jan to 1st Week in Feb:
Generalize our API for any HTML, Javascript, and PHP website

2nd Week in Feb to 3rd Week in Feb:
Usability testing, documentation, and increasing usability for API

4th Week in Feb to 1st Week in Mar:
System testing and any final modifications that need to be made before initial roll out

2nd Week in Mar to 3rd Week in Mar:
Integrate API into an actual website's credential system (Sparespace)

5 Risks

Unacceptable classification time is a major risk to our project. Our current plan is to perform the classification task on the backend server of the developer using our application. We have decided this is the best option, given the alternative of performing the classification on the local user's machine, because professional server's are designed for large amounts of calculation. However, we still have a risk that the backend server may not be able to classification tasks in an acceptable time frame. Since classification is computationally expensive, there is a risk that the application will tie up more computational resources than the developer may find acceptable. This may deter developers for adopting our technology in their access management system. Our contingency plan is to perform the classification computations on our own dedicated server if necessary. That way we can design our server to have the necessary hardware to complete the task in an acceptable timeframe. This contingency plan will be activated in the event that the average login time for our system is greater than the average login time for the traditional username/password approach. These times will be determined during our research and usability tests.

Another risk that we have is that the classifier cannot tell the difference the image of a real person and an image of an image of a person. This possibility would compromise the security of our system, as a simple way to gain access to someone's account would be to print out an image of the person and use that for login. Our current plan is to train the neural network to detect images of images and deny access when encountered. If, during our system testing, this proves to be unsuccessful, our contingency plan is to use time series data to verify that input image is not 2 dimensional. This will add a layer of complexity to our project, but will be implemented if our system testing deems it necessary.

6 Additional Considerations:

6.1 Things We Learned:

6.1.1 The importance of Github

At the start of the semester, we only really used Github as a dump for any code or research we have done. But as we started to get into the implementation side of things, we started to see the value of Github as a way to have a uniform code base that we could all work from. This is not to say we did not have any merge errors are there certainly was, but having the transparency of what everyone else was working on was a great boost to our communication and efficiency.

6.1.2 Using Hipchat for communication

HipChat is a platform like Slack, and we used this to share research links, snippets of code, and various other paperwork that we needed to work on together.

6.2 Things to Improve On:

6.2.1 Using Trello to keep track of sprints

As we went on through the semester, we started to get a better picture of how we were going about the project, and we plan on using Trello more extensively for the next semester to assign tasks for each sprint.

6.2..2 Carefully documenting work

The code we have currently is not as well documented as we would like, and moving forward we will have to work on documenting our code better since we are providing an API as well as having to unit test our code.