ACADEMIC YEAR 2019

KEIO UNIVERSITY

FACULTY OF ENVIRONMENT AND INFORMATION STUDIES

TAKEDA LABORATORY

Behavioral Modeling: Authentication for the Web

Aaron Tan

Abstract

Users typically do not use cryptographically strong passwords. They typically also re-use the same password across multiple accounts and services. Some tools such as 2FA and password managers attempt to provide security by handling password generation or adding an additional step to the login process. Both these methods however involve a degree of additional user input. This leads to a low adoption rate for these tools amongst users. Tools such as Google's reCAPTCHA attempt to secure the login process from automated bots at the cost of user privacy. This undergraduate thesis proposes a system that provides additional security while minimizing user friction for the login process of websites. The authentication system identifies users via their login behaviors to determine if the individual logging into an account was the owner of the account. The rate of authorization for benign logins in which the owner logged into their own account was 72%. The rate of deterrence for preventing unauthorized logins by attackers using the same credentials was 100%.

Acknowledgements

I would like to thank the following people for their support across these four years:

- My family for their constant love and understanding.
- Korry Luke, Bevan Lee, and Lena Kuramochi for pushing me to be a better person.
- Bradley Suzuki, Wei Chi Lin, and Yuka Noda for their patience.

Aaron Tan

Faculty of Environment and Information Studies

Keio University

Table of Contents

1. Introduction

Growing reliance on web services means an increase in the need to create online accounts for each service utilized. Studies suggest the average user has at least 90 accounts [1]. As a username password pair is currently the most common method of authentication, a user could be expected to manage many different credentials while also keeping track of the individual service they belong to. Realistically however, users reuse passwords. This presents a security hole that can be exploited should one of the accounts using shared credentials be compromised. Several tools exist to help users generate secure passwords or provide additional security to their accounts, however these methods require action on the user’s behalf.

1.1 Password Usage

The general user has not proven to be exceptionally security conscious when dealing with online matters. The average person reuses the same exact password an estimated four times [1]. This does not include however, variations of the same password that may simply change a character or append the name of the site it is being used for. Therefore the total amount of reused or similar passwords may exceed the estimated value many times over.

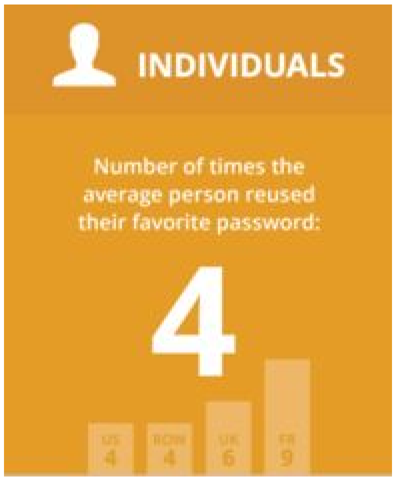
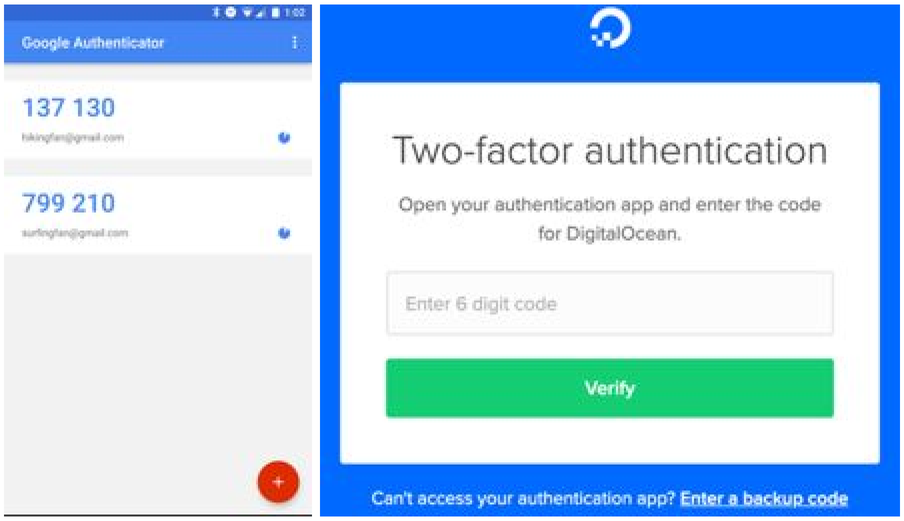


Fig 1. Infographic by password manager company Dashlane [1]

Should any account in the network of shared passwords be compromised, the other accounts within the network are likely to be compromised as well. While educating users on security best practices and password safety may be a viable strategy in ensuring a *company's* employees maintain a level of security awareness, it comes with great monetary cost. Furthermore, employees may hold a high level of security during business hours, but become more lax during personal time. Despite the existence of tools and training designed to improve user security, there is a clear need for server-side protection against such scenarios

1.2 Two-Factor-Authentication

One countermeasure against weak passwords and their re-use is Two-factor-authentication (2FA). 2FA adds an additional layer of security typically be requesting a randomly generated token be submitted. By adding an additional step to the login process, 2FA attempts to protect users from an account breach even under the circumstances with which their credentials have been stolen. The advantage of 2FA is that the generated token is typically only accessible either physically via the user's phone or through a secondary account such as the user's email. Thus forcing the attacker to either physically require access to the user's device or access to their email [2].



Two-factor authentication

Open your authentication app and enter the code for DigitalOcean.

Can't access your authentication app? Enter a backup code

Fig 2. Google's Authenticator App and a 2FA request from Digital Ocean

Where 2FA fails is its adoption rate amongst users: Less than 10% of google accounts utilize 2FA [3]. This may be because it is an additional step that users must deal with before being able to login. It could be thought to be too troublesome for the perceived minimal benefit it provides. 2FA is typically the last barrier an attacker must face to gain access to a web account. In the event that 2FA is compromised, the account is typically breached.

1.3 Password Managers

Password managers are another set of tools used to combat weak and re-used passwords. Password managers are applications developed to securely store username and password pairs along with the service they are used for in order to alleviate the need to memorize credentials or store them in plaintext. Alongside the secure storage of credentials, password managers also provide the feature of generating secure passwords and handling auto-filling input fields with the necessary credentials. The issue with password managers is the same as with Two-factor-authentication, the adoption rate is low amongst users. Only around 12% of Americans use a password manager [2][4]. While password managers make attempts to streamline the process of logging in by auto-filling input fields, they present yet another application that needs to be managed for each login process. While many of the leading password managers provide a free tier for their users, some features such as cross-device syncing are locked behind a subscription paywall. Like 2FA, password managers may be seen to require too much effort in relation to the risk it potentially mitigates.

1.4 Goal

This research proposes a comprehensive secure authentication system with the goal of mitigating poor user security. Users are not safe with passwords and thus a method to differentiate between authorized and unauthorized human users is needed. In the event a login action is processed, behaviors such as the mouse speed and typing speed of the account creator will be used to develop a benchmark which will then be measured against for subsequent login actions. This authentication system aims to protect users from account breaches in the event that their credentials are stolen. It should serve as an invisible layer of security that is not noticeable on the user's end. Furthermore, it should maintain its security even in the event that 2FA is breached.

2. Background

2.1 2FA Issues

In a study examining 100,000 google accounts it was found that only 6.2% of these accounts had Two-Factor-Authentication enabled [3]. As discussed previously, one explanation behind the low adoption rate of 2FA is the extra step required from users to log in. After inputting their login details, users will typically need to either open an additional application, check their phone's SMS messages, or look at their email accounts for the 2FA code. Users that value their security and privacy may be willing to go through with the extra step. The majority of users however, will likely opt for a smoother login experience by opting not to activate 2FA. Two-factor-authentication is being made further less effective with the rise of phishing attacks. Imposter websites that trick the user into submitting their credentials can also ask for a 2FA code. Furthermore, the Password Reset Man In The Middle attack demonstrates the ease with which Two-Factor-Authentication can be bypassed [5]. As such, it is important to consider alternative measures of securing login processes that maintains a comfortable user experience.

2.2 Data Modeling

Profiling users by measuring how they interact with a system is prevalent. Such systems are typically set in place to provide security in some form. Police Agencies may profile citizens in an attempt to detect criminal behavior or patterns. Software may implement an intrusion detection system(IDS) that monitors behavior to detect malicious attac. Websites however are known to profile their users to provide targeted ads by selling their data to 3rd parties. Similarly to an IDS, this study proposes a system that aims to gather behavioral information in an attempt to prevent unwanted access to a resource. The main difference is the environment in which each system exists. Intrusion detection systems largely target either network layer attacks or monitor command line inputs. This study however proposes an authentication system that can operate on a website. Recent events such as the Facebook Cambridge Analytica have led to an increase in awareness regarding one's online identity [6]. It is important that this authentication system gathers only the bare minimum data from its users. This study first examines past research conducted on IDS' to form its methodology.

Traditionally, data such as typing speed, the type of commands sent, session activity, and device information is gathered for the purposes of intrusion detection. Targeting GUI applications allows for an IDS to capture additional data that a command line or shell based intrusion detection system would not be able to capture. Data such as mouse movement activity adds another factor that may be used when monitoring GUI applications for malicious activity. Depending on the implementation, this data model may be linked to a browsing session if the system is a shared environment, or to an individual account. [7] Using the methodologies of existing intrusion detection systems, this research’s authentication system can employ similar techniques to effectively profile a users behavior and thus differentiate between an accounts owner and unauthenticated users.

Machine learning algorithms and user modeling brings with it the issue of needing large datasets to accurately predict behavior. Webb et al. suggests that the need for large datasets can be mitigated by increasing the amount of features being measured and decreasing the overall strictness of the measurements. Furthermore, they suggest measuring psychometric data such as user preference [8].

Past studies have attempted user authentication through monitoring specific behavioral factors. Revett et al. focuses on mouse movements as a means of user authentication. While the study suggests that a mouse-based authentication system is viable, their methodology deviates from the standard username password model and opts to rely solely on mouse movement patterns for user authentication. As this research aims to develop a comprehensive authentication system that requires no additional user friction, Revett’s exact methodology was not implemented. It suggests however, that mouse based movements can be used for behavioral analysis. Mouse movements are therefore one of the factors being measured for this authentication system [9].

Haider et al. studied how keystroke dynamics could be analyzed for user identification. Their findings suggest that individuals are able to be profiled through their own unique typing signature akin to biometric measurements. Their methodology of measuring the individual timings in between keystrokes typed will be used as the basis of this thesis' data model [10].

2.3 reCAPTCHA

Concerns regarding privacy and user experience contributed towards the development of this study. Developed mainly to combat cross site request forgery and other automated attacks, reCAPTCHA is a service created and maintained by Google. reCAPTCHA promotes three main tenants. Providing advanced security to prevent spam and abuse, keeping a consistent and easy-to-use service, and using the data it collects to further machine learning and AI research. On each request made to a site, reCAPTCHA monitors the user's interactions in order to generate a score between 0.0 and 1.0 representing the likelihood of the interactions having originated from a bot or human respectively. Depending on which version of reCAPTCHA is being used, the user may be asked to solve a challenge before proceeding with their request. reCAPTCHA has gone through three major iterations, each aiming to improve upon its three tenants.

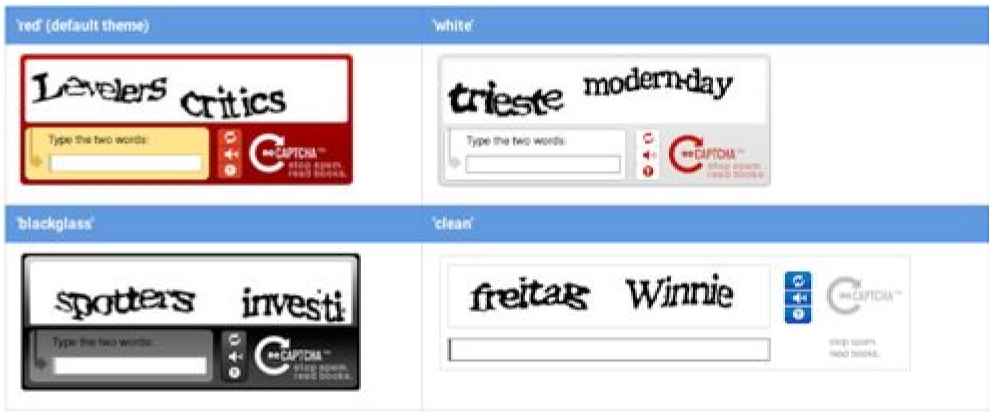


Fig 3. v1 demonstrating a text distortion challenge

reCAPTCHA v1 used distorted text as its challenge. Users were asked to type the displayed text in the input before being able to submit their form data. Due to the nature of the distorted text, it was difficult for bots to enter the expected text. The data gathered by reCAPTCHA v1 was used to improve computer vision technologies such as optical character recognition.

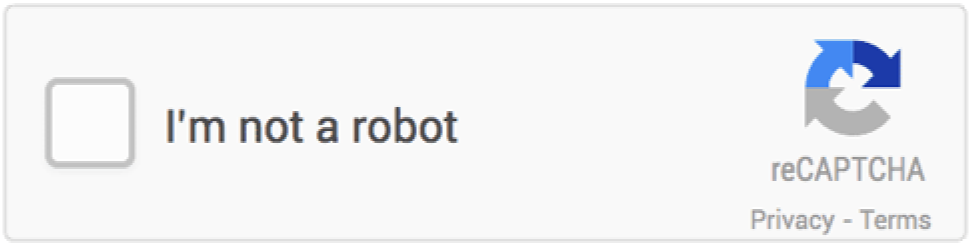


Fig 4. v2 collecting data before deciding to display a challenge

reCAPTCHA v2 improved upon this method by utilizing images captured from Google Streetview. V2 asked users to click a button and by measuring attributes such as mouse speed, would determine whether or not to display the challenge or simply allow the user to pass. The challenge presented would be a Google Streetview image divided into a grid. Users would then be asked to identify specific items within that grid. reCAPTCHA v3 puts the handling of presenting a challenge or any other action in the hands of the developers and instead simply monitors a webpage for suspicious activity.

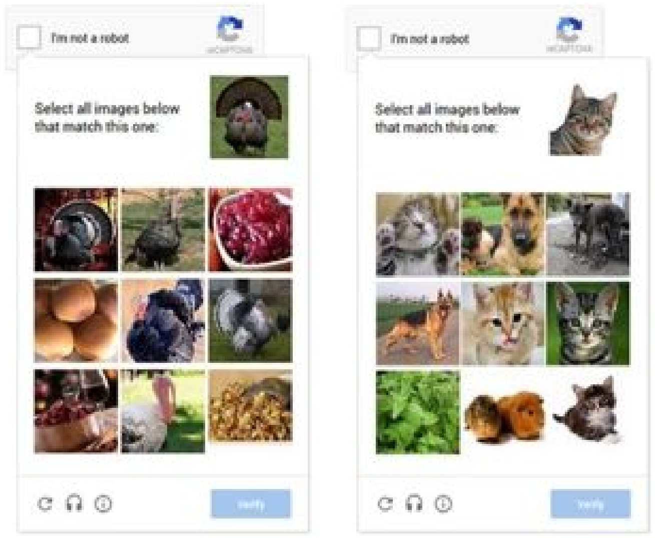

I'm not a robot
Select all images below that match this one:
Verify
I'm not a robot
Select all images below that match this one:
Verify

Fig 5. v2 displaying a challenge

reCAPTCHA is designed to differentiate between when a human decides to access a web page and when a bot or automated process does the same. During the login process, sites that implement reCAPTCHA will typically present a challenge to the user as a means to prevent against bots. In the event that the challenge is failed, users are asked to continue clear newly generated challenges until the user either clears the challenge or their score as measured by the system improves past the threshold set by the developer. This introduces friction during the login process. In an ideal scenario, users should not need to solve challenges at all, the underlying system should be able to determine between bots and users.

reCAPTCHA v3 attempted to solve this issue but information regarding the data that it gathers and how that data is used has been vague. Google encourages site administrators to add reCAPTCHA not only on login pages or pages with user input but on all pages of one’s site. There have been concerns regarding if this allows Google access to a user’s browsing habits within a reCaptcha v3 powered site.

As opposed to reCaptcha’s focus on bots, this study attempts to solve the issue of malicious login attempts made by humans. The authentication system aims to only gather the minimum data required to accurately identify a user without compromising their privacy. [11]

3. Design

This resume proposes a method of distinguishing between different human users by monitoring their log in behaviors with the goal of developing an effective authentication system. To distinguish between the account owner and unauthorized login attempts, a benchmarking system will be used. User data that has been gathered will be processed to develop benchmarks which further login attempts must pass in order to be successfully authorized.

3.1 Data modeling

Traditional intrusion detection systems focusing on command line interfaces must take into consideration the large amount of various commands that can be input into a terminal. This study however, focuses only on a login scenario. For each individual user, there are only two expected inputs, the username and password. The restriction on input provides an opportunity to be stricter with the measurements.

The timestamps of each time a key has been pressed is recorded. The authentication differentiates between keypresses made while typing the username and typing the password. This allows data analysis to occur separately between the two inputs. Using the keypress timestamps, three pieces of data are extrapolated.

The simplest data type to extrapolate from the keypress timestamps is the number of keypresses made during a login attempt. users have the option of switching input fields using the tab key or submitting the login form by using the enter key. As user preference for this behavior varies, it is necessary to measure the total number of keypresses made during each attempt.

Typing speed is a common type of data gathered for user data modeling. One method of measuring typing speed is by dividing the time spent typing by the amount of characters or words typed. Due to the nature of the text being measured, that being a password and username. Measuring typing speed by word count is ineffective. This study instead measures by characters per minute (CPM).

Alongside measuring typing speed in CPM, this study will also measure keystroke delta. Keystroke delta is defined as the time in-between keypresses. Each user has a username and password of fixed length. User’s typing behavior also typically remains the same. Haider suggested that a user’s keystroke dynamics are difficult to reproduce [10]. By measuring the keystroke delta, a signature can be developed that is unique to the user.

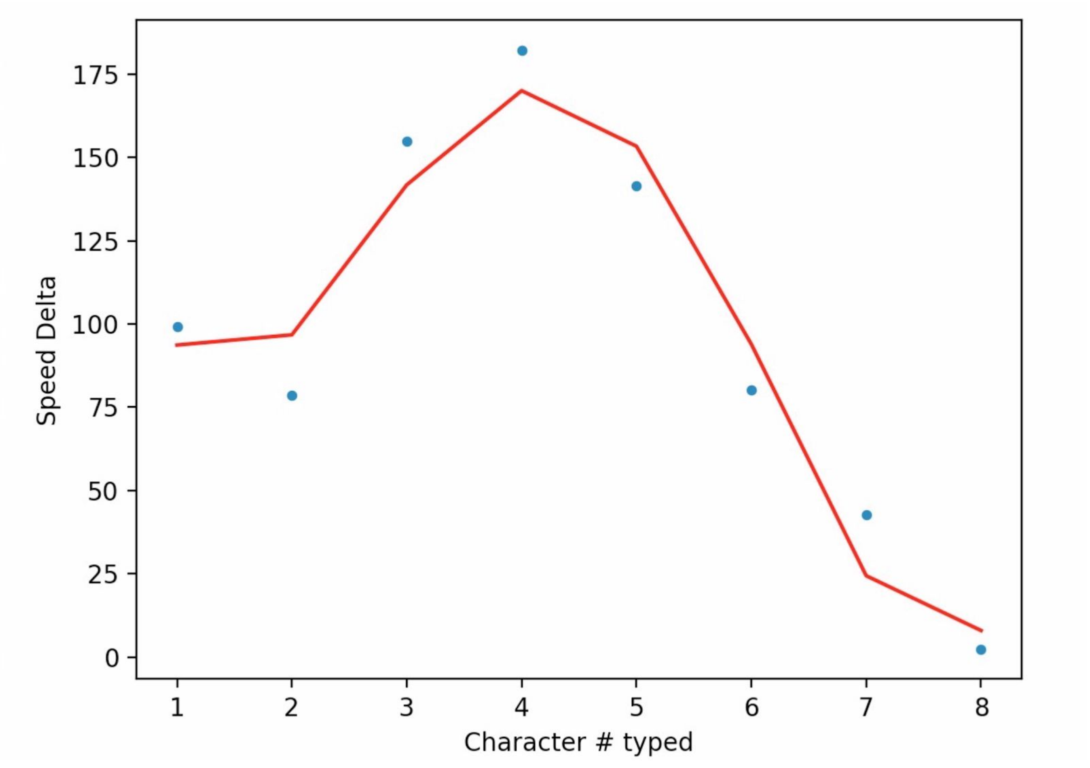
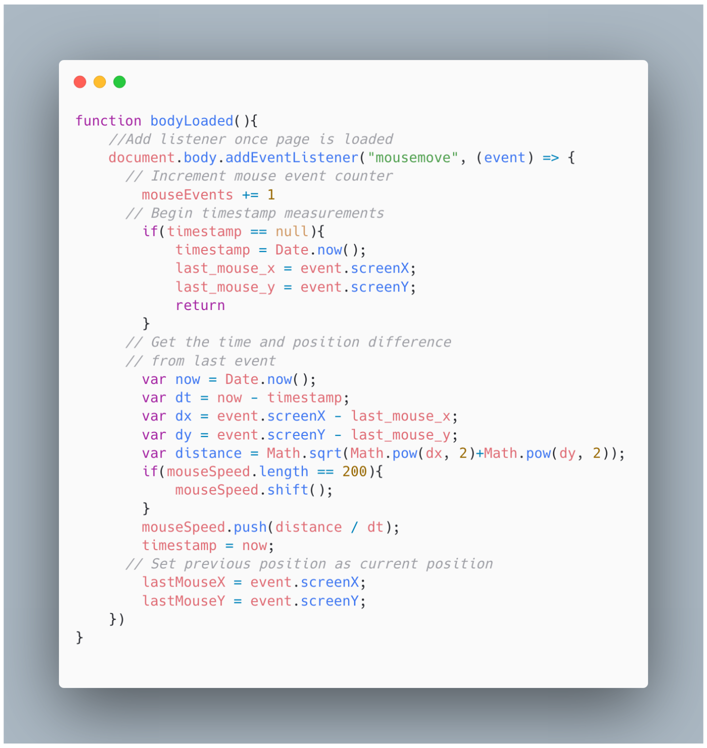


Fig 6. Graph depicting keystroke timings and predicted keystroke delta

Behavior regarding mouse usage varies depending on the user. For this reason, mouse activity is also gathered. Mouse speed measured as the distance between two coordinates divided by the time difference. Alongside speed, the total amount of mouse events is measured. In the event that users attempt a login action on a mobile device or touch-screen device, mouse speed will not be gathered. Screen taps however will be measured as clicks. Javascript is used to capture mouse move events.



```javascript
function bodyLoaded(){
    //Add listener once page is loaded
    document.body.addEventListener("mousemove", (event) => {
      // Increment mouse event counter
        mouseEvents += 1
      // Begin timestamp measurements
        if(timestamp == null){
            timestamp = Date.now();
            last_mouse_x = event.screenX;
            last_mouse_y = event.screenY;
            return
        }
      // Get the time and position difference
      // from last event
        var now = Date.now();
        var dt = now - timestamp;
        var dx = event.screenX - last_mouse_x;
        var dy = event.screenY - last_mouse_y;
        var distance = Math.sqrt(Math.pow(dx, 2)+Math.pow(dy, 2));
        if(mouseSpeed.length == 200){
            mouseSpeed.shift();
        }
        mouseSpeed.push(distance / dt);
        timestamp = now;
      // Set previous position as current position
        lastMouseX = event.screenX;
        lastMouseY = event.screenY;
    })
}
```

Fig 7. Demonstration of mouse movement gathering function

Browser and OS data will also be gathered. Browser data exists as an additional factor to assist in differentiating users. However as the vast majority of users use either Firefox, Google Chrome, or Internet Explorer, this data will not be weighted as heavily. OS data however, will be used to separate datasets for users that operate on multiple different devices by saving the data as a new document in a different collection within MongoDB. In the event that the collection lacks sufficient data to effectively distinguish between authorized and malicious login attempts, the existing datasets will be used. Separate datasets are necessary as using a different device typically means using a different keyboard and mouse. Changing these devices may affect the user’s behavioral patterns. As such, a distinct dataset needs to be created to record the data that has been affected by these changes. Future logins using that device will then be compared against the appropriate dataset.

Data type	Details
Typing Data	Time stamps
Mouse Data	Time stamps, coordinates, move events
Browser	Distinguishes only type and not version
OS	Used for dataset comparison

Fig 8. Types of data gathered directly from user

Browser fingerprinting is the process of generating a unique fingerprint for each browser that allows for a high amount of differentiation and tracking to occur. Browser fingerprinting typically relies on extensive gathering of data that crosses the line of what this study deems acceptable. Websites such as www.amiunique.org [12] and studies such as Eckersley [13] demonstrate how easy it is to track a user based on various tricks such the device's screen size and browser window orientation. To preserve user privacy, libraries such as FingerprintingJS2 [14] have remained intentionally unused.

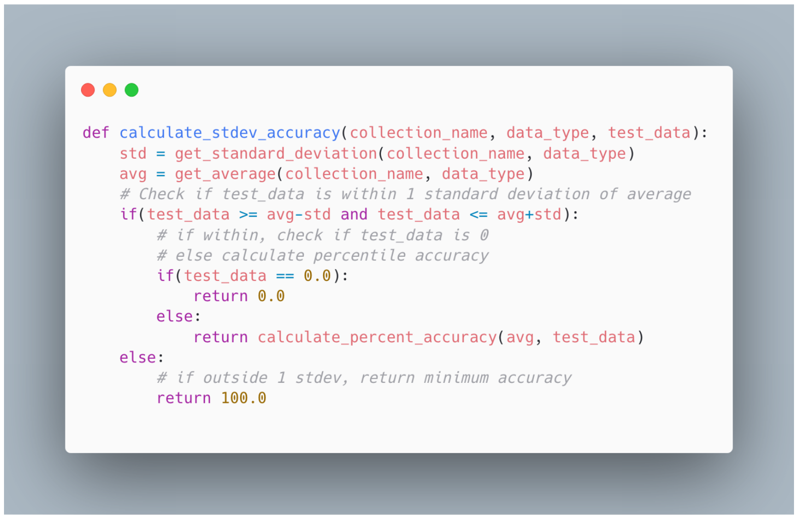
3.2 Developing Benchmarks

On each login attempt, two sets of data are used. The data gathered during the login attempt will be referred to as test data. The data gathered from the training phase and past successful logins will be referred to as training data. Depending on the type of data within each dataset, the method of analysis is different. Despite using different methods of statistical analysis, every data type compared results in a value between zero and one hundred representing how far the test data was to the training data. A value of zero indicates the result was perfectly accurate.

Data Type	Analysis Method
Username Typing Speed	Standard Deviation + Percentile Accuracy
Password Typing Speed	Standard Deviation + Percentile Accuracy
Username Keystroke Delta	Polynomial Regression + Percentile Accuracy
Password Keystroke Delta	Polynomial Regression + Percentile Accuracy
Total Keypresses	Standard Deviation + Percentile Accuracy
Mouse Speed	Standard Deviation + Percentile Accuracy
Mouse Move Events	Standard Deviation + Percentile Accuracy
Browser	Standard Deviation + Percentile Accuracy
OS	N/A

Fig 9. List of all processed data types and their analysis method

The typing data is measured by comparing whether or not the test data falls within a standard deviation of the training data. If the test data falls within one standard deviation, the authentication system then measures the percentile accuracy of the test data to the mean. If the test data does not fall within one standard deviation, the percentile accuracy value is one hundred.



```python
def calculate_stdev_accuracy(collection_name, data_type, test_data):
    std = get_standard_deviation(collection_name, data_type)
    avg = get_average(collection_name, data_type)
    # Check if test_data is within 1 standard deviation of average
    if(test_data >= avg-std and test_data <= avg+std):
        # if within, check if test_data is 0
        # else calculate percentile accuracy
        if(test_data == 0.0):
            return 0.0
        else:
            return calculate_percent_accuracy(avg, test_data)
    else:
        # if outside 1 stdev, return minimum accuracy
        return 100.0
```

Fig 10. Calculation of standard deviation accuracy and handling of mouse cases

Mouse related data types such as mouse clicks, mouse speed, and mouse movement are a special case. Users are able to navigate the login page and submit a login attempt using only the keyboard. In the event this happens, no mouse related data would be captured. To compensate for instances in which this occurs, if the mouse data is zero but still within the standard deviation, the percentile accuracy will be zero. Cases in which mouse data is not zero will use the same comparison method as typing data.

As keystroke delta is the most identifiable type of data being gathered, a more precise method of analysis is used. Keystroke delta training data is processed using polynomial regression. Polynomial regression is a machine learning algorithm that attempts to plot a line of prediction with more accuracy than linear regression. Using polynomial regression, a list of prediction values is generated. The values correspond to the predicted amount of time in-between each keystroke. Mean percentile accuracy between the keystroke delta test data and predicted keystroke timings is then measured.

After each piece of data has been processed and analyzed, the authentication system is then begins evaluating whether or not to authorize the login attempt. Each data type analyzed results in a value that represents the accuracy of the test data to the training data. The authentication system compares the accuracy to an expected accuracy value. If the expected value is met or exceeded, the data type passes the evaluation.

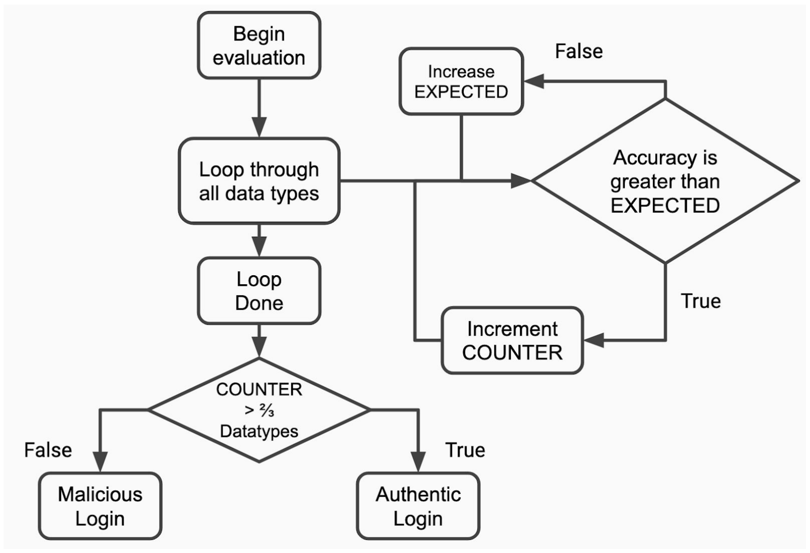


Fig 11. Diagram of evaluation flow

If a data type does not meet the expected accuracy, the data type fails the evaluation, this causes the baseline expected accuracy to increase for all data types. Upon one data type failing to meet accuracy, the strictness by which all other data types are being measured is increased. A majority of data types must pass the evaluation for the authentication system to authorize the login attempt. This system of authentication ensures that login attempts are flexible enough to account for the account owner’s human error. Should the account owner fail the keystroke delta evaluation, they are not immediately rejected. Instead, further evaluations on data types such as typing speed and mouse movement are made more strict. If the user’s inputs pass the evaluation of the stricter tests, they will still be granted access should the proper amount of tests pass.

3.3 Testing

The data gathering is split into three phases. These phases are in order, the training phase, the testing phase, and the attacking phase.

During the training phase, volunteers are shown their generated username and password credentials. They are asked to then contribute to the training dataset by successfully logging into the authentication system twenty-five times. The first five login attempts are then removed to account for the users’ unfamiliarity with their credentials.

Once the testing phase begins, users are asked to login with the same credentials presented to them in the first phase. Upon each login attempt, the authentication system will respond with a screen indicating either success or failure. During this phase, the number of successes and failures will be recorded for each volunteer.

In the attacking phase, each volunteer will be randomly assigned the credentials of a separate volunteer and asked to login using the newly assigned credentials 20. Volunteers are instructed to login to the account with a variety of behaviors. These login attempts are intended to mimic a malicious attacker trying to access an account. The success and failure rate of these attempts is recorded.

4. Implementation

For the purpose of gathering data, a website was created to host the login authentication system. The system is comprised of a frontend that is monitored, and a backend that handles the processing and storage of any data that is gathered. The frontend for this authentication system is a web page that users may login to using a username password pair. Upon entering their credentials, the user will be brought to a results page that displays whether or not the login is a success or a failure based on how their behavioral data is compared to the data stored on the server. This web page will serve as the platform on which tests will be conducted and demonstrates how the system operates.

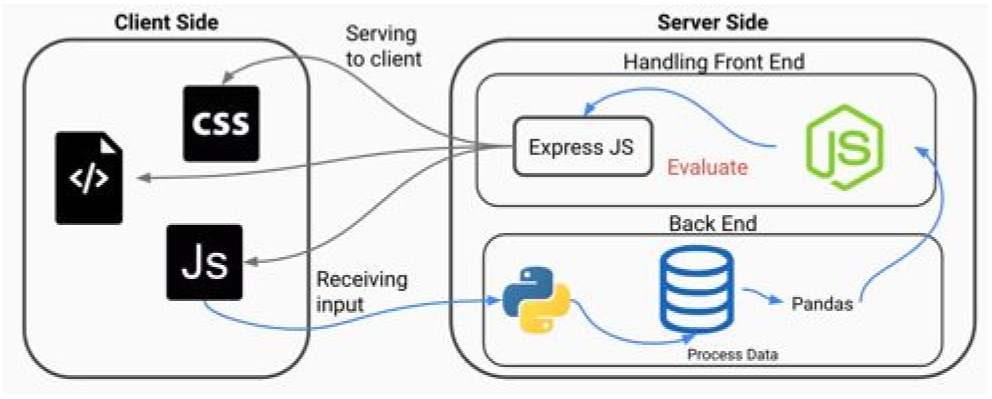


Fig 12. Overall flow of Authentication System

4.1 Frontend

NodeJS [15] will be used as the main frontend language and runtime. It was chosen as it offers a wide array of web-based packages that can be used to quickly build a website that handles routing, rendering, and authentication. The main package used to develop the frontend is ExpressJS [16].

ExpressJS is a web framework that can be used to build web applications and APIs. ExpressJS will handle the routing of the web page. This involves displaying the login page and results page. The login page will contain a form that sends a request to the backend for validation. Within the request body are the login credentials and the data gathered from that particular login session. If the credentials are validated, a second request will be made to validate that the individual submitting the credentials is the expected individual. The handling of all request validations will be performed by the backend such that the user has no ability to affect the validation process.

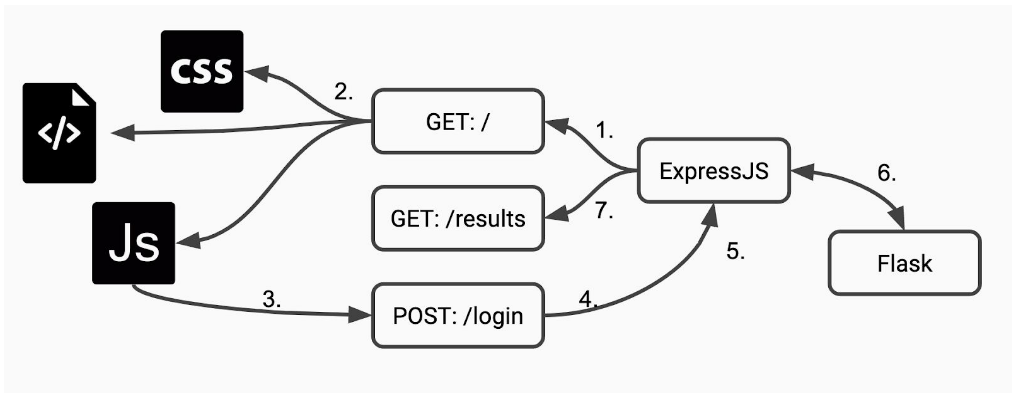


Fig 13. Flow of frontend

Figure 6 demonstrates the flow of the frontend in 7 steps. First, ExpressJS listens for GET requests at the root route ‘/’. Once a GET request is made, the HTML, CSS, and JS files are served to the client. When the client logs in, the JavaScript performs a POST request to the ‘/login’ route. ExpressJS receives this request and makes a secondary request to Flask, a Python based web framework, for data processing [17]. Once the backend finishes processing the data, Flask sends a response to the request made from ExpressJS, which then renders the '/results' page with either a success or failure screen.

4.2 Backend

Python will be used as the handler for data processing. It was chosen due to the ease with which data object generation and manipulation can be achieved.

MongoDB is a document-based NOSQL database that stores its data objects in a JSON-like format [18]. MongoDB's data model is separated into collections and documents. Collections are groupings of documents. A document is a data object that stores key-value pairs. The values store themselves can be data objects as well such as strings and arrays. This allows for ease in data modeling across both Python and NodeJS, two languages that handle data objects in a JSON-friendly manner. Choosing a NOSQL database also provides flexibility in designing the database schema. A traditional SQL database requires defining all data types beforehand. As this thesis continuously iterated upon the existing schema to add and remove data types, NOSQL proved to be a better fit.

In order to receive data from the front-end and transmit processed data back, an API is built using the Flask module. Similar to ExpressJS, Flask is a minimal web framework module for setting up a web application or API.

While Python can be used to quickly create and manipulate data objects, it lacks the ability to easily perform complex analytical operations. Various libraries such as Numpy[19] and SciKit-Learn[20] will be used to facilitate the data analysis.

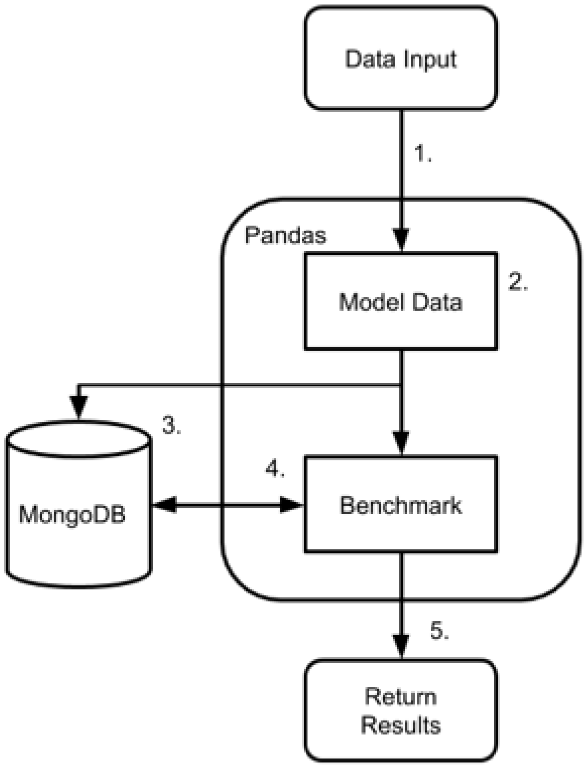


Fig 14. Flow of backend

Figure 7 details the flow of the backend in 5 steps. The backend first receives login attempt details and measured data from the frontend as input. The data is processed into a format for ease of analysis. A query is then made to MongoDB for the historic data of the user attempting to log in. Once MongoDB fetches the data, it is then compared with the input data. After comparing against the benchmarks, in the event of a successful login, the data is stored within MongoDB and the results are output. Details about how the benchmarks are compared against input data will be discussed in the implementation section.

4.3 Generating User Credentials

The system tests user login behavior against past historic behavior by the same user. To generate this data, users are asked to repeatedly manually login to the account. This means they are required not to copy paste or use 3rd-party applications such as password managers to fill the username and password forms for them. To preserve the privacy and security of users, they will be asked not to use their own password details. Instead, accounts will be prepared ahead of time and assigned to each user.

Random entries selected from the word list will be concatenated to form username and password pairs. This ensures that test users will have unfamiliar passwords. Using this method also helps a user type closer to their normal speed as opposed to if the username and passwords were randomly generated strings of individual characters, numbers, and symbols.

No API was found that provides an interface for generating random dictionary words. Instead it was decided to use a wordlist of the top 1000 passwords[21]. This presents a more realistic scenario to test against as the system will need to develop unique benchmarks against real passwords as the input.

4.4 Data Gathering

This undergraduate resume will be evaluated based on how accurate the authentication system developed can identify between account owners and attackers. However, it should also be able to distinguish between account owners and automated login attempts conducted by bots. To ensure user privacy, an attempt was made to gather only non-identifiable data. This data is defined as characteristics of an individual that cannot be easily be used to identify the individual [22].

Volunteer users will be gathered to provide the authentication system with the data necessary to conduct testing. Volunteers will be selected under the criteria that they are fluent in English and are comfortable typing in English as well. Volunteers will be briefed regarding the nature of the tests and are given the option to leave the study at any time. They will be notified regarding all types of data being gathered during the tests.

The data gathered is as follows and is only applicable on the login page:

- Timestamps of all keypresses made
- Coordinates and timestamps of all mouse movements
- Browser Type
- OS Type

Mobile devices present the opportunity to gather further information such as location, accelerometer as well as biometrics. Despite the accuracy these factors may contribute to the authentication system's measurements, they have been considered out of the scope of this study. As one of the goals of this study is to preserve the privacy of users, these will not measured [23]. Mouse movement will also be ignored for mobile users. The data of each successful login will be saved to the MongoDB database as an individual document.

After sufficient successful logins have been stored within the database, the authentication is ready to process data to compare against malicious login attempts. Users will be given the credentials of other existing users and asked to attempt to log in. To ensure that false positives or successful logins do not contaminate the existing dataset during the testing phase, the database will be rolled back to its previous state before each login attempt.

5. Experiment

As discussed in chapter 3.3 Testing, volunteer users were gathered and asked to participate in the experiment. Three phases of testing were conducted that served to first train the dataset with positive login training data, then test the authentication systems account owner recognition rate, and finally test the systems attacker deterrence rate. The purpose for these tests is to measure how effectively the authentication system is able to work as an invisible layer of protection for web logins without compromising user experience. In an ideal scenario, the authentication system should successfully identify when the account owner logs in with their own credentials with a rate of 100% accuracy. Similarly, the authentication system should prevent all malicious attacks wherein an attacker logs in with the same credentials with 100% accuracy. This research aims to reach at least 80% accuracy for both the owner identification and the attacker deterrence metrics.

5.2 Data

This study measured the rate at which users were able to login to their own account to first measure the accuracy with which the authentication system was able to identify the account owner's login attempts. Ten volunteers were given accounts and trained the dataset with twenty login attempts. They then attempted to login twenty times and the system evaluated their inputs. The following figure demonstrates the successful and failed logins across the 20 attempts. Logins in which the credentials were incorrectly input were not counted.

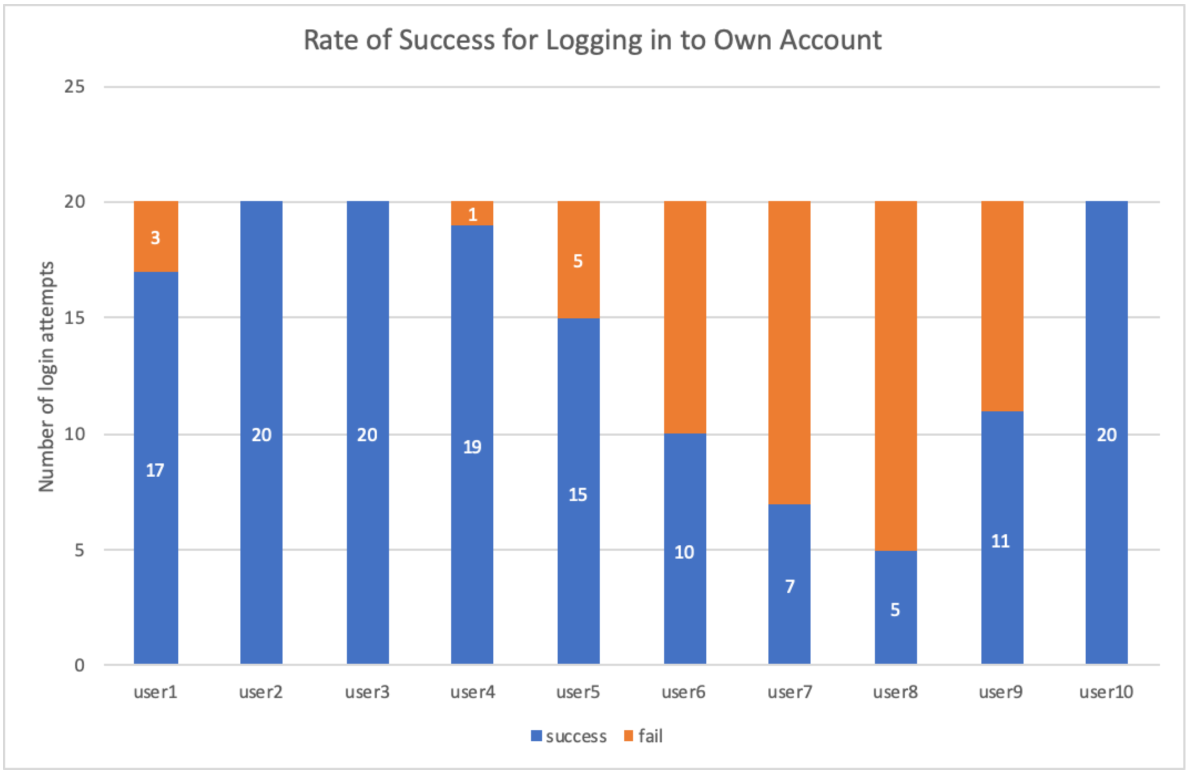


Fig 15. Graph demonstrating success to failure rate of login attempts as owner

The third phase asked volunteers to assume the role of an attacker. They were given the credentials of a random user and tasked with logging in 20 times to mimic a malicious user attempting to gain access to the account. Their success rate for logging in as an attacker was measured.

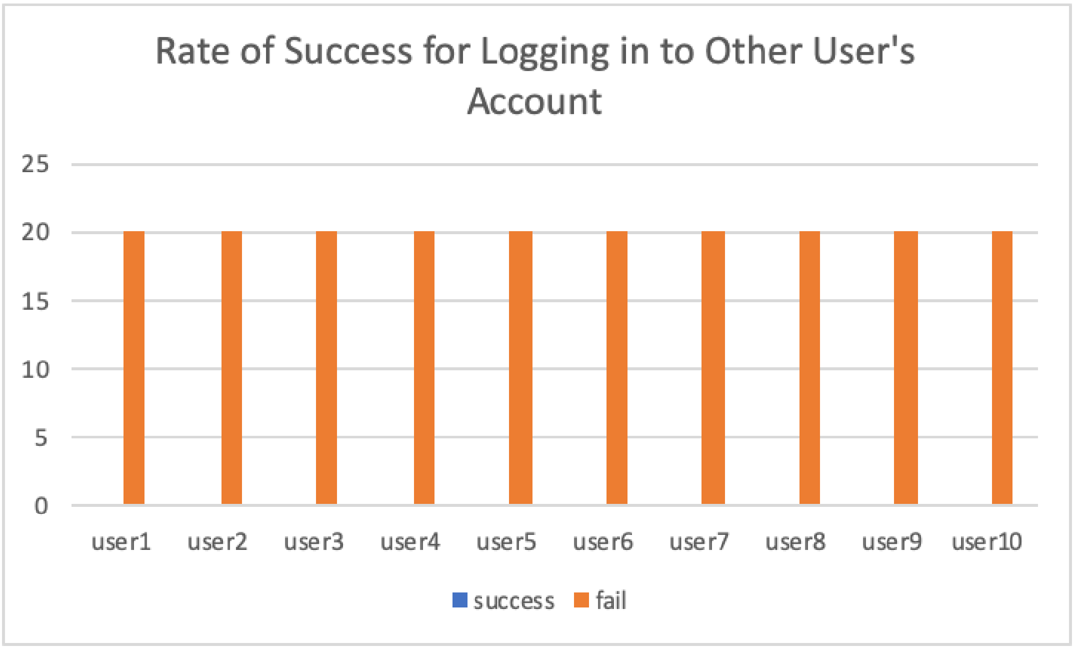


Fig 16. Graph demonstrating success to failure rate of login attempts as attacker

5.3 Success rate

There are two statistics that are important for this study. The success rate of login attempts as the owner, and success rate of login attempts as an attacker. The first statistic demonstrates how effective the authentication system is at identifying the owner of an account. If the owner is unable to login to their own account at a high enough rate, the system fails its core purpose as an authentication system. The second statistic demonstrates the effectiveness of the authentication system in deterring malicious login attempts.

	Mean Success Rate
Owner Login Attempts	72%
Attacker Login Attempts	0%

Fig 16. Table with values representing mean success rate of owner and login attempts

6. Conclusion

6.1 Results

While the data representing owner login attempts appears valid, the data for attacker login attempts raises concerns. Across all volunteers, none were able to log in to another volunteer's account within 20 attempts. There are various explanations that can be used to explain the abnormal success rate. Due to the low number of volunteers, it is likely there is not enough data to accurately represent the system. Further testing must be conducted with a larger number of volunteers. Furthermore, the system's strictness regarding how closely test data must be to training data may need to be adjusted. Currently, the underlying system for gathering user data is present. The analysis that occurs server-side uses polynomial regression, standard deviation, and percentile accuracy to evaluate the test data. The data is compared to an expected value. The analysis method or expected value may need adjustments to finetune the resulting accuracy of the system. As it stands, the authentication system is too strict. This has lead to the owner success rate being low. It can be expected however, that decreasing the strictness of the system to increase owner login success rate will also increase attacker login success rate. A balance must be found between the tradeoff of owner recognition and attacker deterrence. Extensive testing will need to be conducted in order to be confident with the accuracy of the system.

6.2 Limitations

The motivation behind this study is to provide security for users during the login process. Current methodologies such as Two-Factor-Authentication and Password Managers exist that require effort by the user in order to be effective. This has led to low adoption rates. Similarly, services such as Google’s reCaptcha are providing a method to prevent unauthorized automated activity. This comes at the severe cost of privacy. Targeting users that forgo these tools and services however means that the authentication system is unable to respond to those that do use these services. If a user or password manager instantly fills in the credentials, there is no delta time or similar data to gather. To alleviate this issue, sites that implement the methodology presented in this study should ensure that authentication via user behavioral data is a togglable feature. Users that wish to enable it should be notified to not use auto-fill or to copy and paste their credentials.

Users should always have the ability to change passwords. Whether the cause is due to human forgetfulness or security concerns, the password should still be able to be changed. However the current design of the authentication system focuses on a dataset built off a specific, fixed-length password. The dataset would need to be re-generated in the event that the password is changed. This presents a flaw in the security of the authentication system as an untrained or new dataset would lead to less accurate measurements and possibly allow attackers access. As future research, observing the user’s keystroke biometrics across a dynamic-length string could lead to a system that is able to predict the user’s typing signature across all possible password inputs.

The decision to save the individual delta time between keystrokes for each user may prove detrimental in the event that the system is compromised. In a practical scenario, passwords are properly salted and hashed. However, storing the delta times are explicitly related to input length. Should the system be compromised in any way, the delta times will reveal the length of each user's password.

As the authentication system relies on data gathered client-side that is then sent to the server, it is possible for a MITM attack to intercept a login attempt with data values that pass all tests. They can then recreate the login attempt to gain access to the data. Future studies have the option of disabling data inputs in which the request contains the exact same data as an entry that already exists within the database. This approach would fail however if the attacker is able to modify the packet or decide to manually attempt to recreate the attack.

It must be noted that attacks wherein the attacker has personal connections to the victim or through reconnaissance is able to derive the victim's behavioral patterns would lead to an increased chance that a malicious login attempt is allowed. Knowledge of what data is being gathered will allow attackers to impersonate the behavior of their victims in an attempt to bypass the security measures. Furthermore, while some factors measured such as keystroke delta are more unique, other factors such as typing speed can be shared across multiple users. It is the combination of all the factors that adds strength to the system. It is therefore important that users remain consistent in their behavior.

6.3 Future Research

Implementing a system that integrates behavioral data as part of its analysis opens up possibilities for examining how intentionally altering one’s behavior during login procedures could offer additional security against potential attackers. Knowledge of how the system measures behavior could lead to users delaying certain keystrokes or changing the speed of their typing in order to generate a behavior profile unique to individual websites. As this study focuses on securing the general user without need for additional input from their part, the aforementioned tactics were not investigated. As profiling becomes more intrusive however, such strategies may need to be employed to retain user privacy.

This study aimed to build a system able to differentiate between human users for the purpose of authentication on a website. The target accuracy of 80% acceptance however was not reached. Further studies may be conducted to build upon this study by adjusting the benchmarking system. For this purpose, the source code can be found at https://github.com/aa-tan/gp.

7. References

[1] Bras, T. L. (2015, July 21). [INFOGRAPHIC] Online Overload – It's Worse Than You Thought. Retrieved from https://blog.dashlane.com/infographic-online-overload-its-worse-than-you-thought/

[2] What Is Two-Factor Authentication (2FA)? (n.d.). Retrieved from https://authy.com/what-is-2fa/

[3] Milka, G. (2018, February 21). Retrieved January 16, 2019, from https://www.youtube.com/watch?time_continue=432&v=W2a4fRalshI

[4] Petsas, T., Tsirantonakis, G., Athanasopoulos, E., & Ioannidis, S. (2015, April). Two-factor authentication: is the world ready?: quantifying 2FA adoption. In Proceedings of the eighth european workshop on system security (p. 4). ACM.

[5] Gelernter, N., Kalma, S., Magnezi, B., & Porcilan, H. (2017, May). The password reset mitm attack. In Security and Privacy (SP), 2017 IEEE Symposium on (pp. 251-267). IEEE.

[6] Lapowsky, I. (2019, March 18). How Cambridge Analytica Sparked the Great Privacy Awakening. Retrieved from https://www.wired.com/story/cambridge-analytica-facebook-privacy-awakening/

[7] Garg, A., Vidyaraman, S., Upadhyaya, S., & Kwiat, K. (2006, April). USim: a user behavior simulation framework for training and testing IDSes in GUI based systems. In Proceedings of the 39th annual Symposium on Simulation (pp. 196-203). IEEE Computer Society.

[8] Webb, G. I., Pazzani, M. J., & Billsus, D. (2001). Machine learning for user modeling. User modeling and user-adapted interaction, 11(1-2), 19-29.

[9] Revett, K., Jahankhani, H., De Magalhaes, S. T., & Santos, H. M. (2008, June). A survey of user authentication based on mouse dynamics. In International Conference on Global e-Security (pp. 210-219). Springer, Berlin, Heidelberg.

[10] Haider, S., Abbas, A., & Zaidi, A. K. (2000). A multi-technique approach for user identification through keystroke dynamics. In Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0 (Vol. 2, pp. 1336-1341). IEEE.

[11] ReCAPTCHA. (n.d.). Retrieved from https://www.google.com/recaptcha/intro/v3.html

[12] Node.JS. (n.d.). Retrieved from https://nodejs.org/en/

[13] Express - Node.js web application framework. (n.d.). Retrieved from https://expressjs.com/

[14] Flask. (n.d.). Retrieved from http://flask.pocoo.org/

[15] Open Source Document Database. (n.d.). Retrieved from https://www.mongodb.com/

[16] NumPy. (n.d.). Retrieved from https://www.numpy.org/

[17] Scikit-learn: Machine learning in Python. (n.d.). Retrieved from https://scikit-learn.org/

[18] Danielmiessler. (2019, July 05). Danielmiessler/SecLists. Retrieved from https://github.com/danielmiessler/SecLists

[19] Brownsword R. (2008) Knowing Me, Knowing You – Profiling, Privacy and the Public Interest. In: Hildebrandt M., Gutwirth S. (eds) Profiling the European Citizen. Springer, Dordrecht

[20] Shi, E., Niu, Y., Jakobsson, M., & Chow, R. (2010, October). Implicit authentication through learning user behavior. In International Conference on Information Security (pp. 99-113). Springer, Berlin, Heidelberg.

[21] Learn how identifiable you are on the Internet. (n.d.). Retrieved from https://amiunique.org/

[22] Eckersley, P. (2010, July). How unique is your web browser?. In International Symposium on Privacy Enhancing Technologies Symposium (pp. 1-18). Springer, Berlin, Heidelberg.

[23] Vasilyev, V. (2019, January 08). Valve/fingerprintjs2. Retrieved from https://github.com/Valve/fingerprintjs2

8. Appendix

Source code for the Authentication System can be found at this page:
https://github.com/aa-tan/gp