Measuring user Behavior for a Web IDS

AARON TAN[†1]

Abstract: Users typically do not create strong, hard to guess 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 by challenging users to complete a series of tasks before they are able to log in. This undergraduate resume proposes a system that consolidates the features of these tools to provide an invisible layer of security during the login process. The tool should identify users via their login behaviors to determine if the individual logging into an account was the owner of the account. A threshold-based system was used to determine if the details and method of input during a login process matched those of the account owner.

Keywords: IDS, Profiling, Security, 2FA

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 still 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.

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. Despite the existence of tools designed to improve user security, there exists a clear need to protect 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 one more step to the login process, 2FA attempts to protect users from an account breach even under the circumstance with which their credentials have been stolen.

†1 Keio University

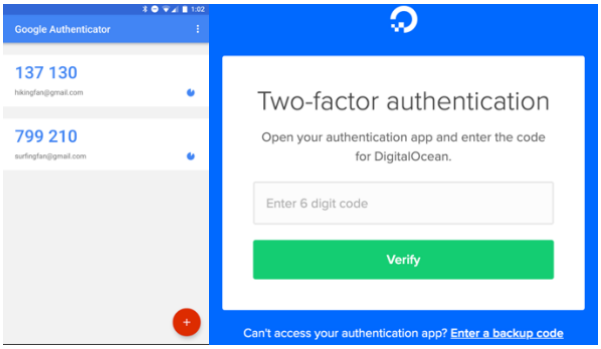

Fig 2. Google's Authenticator App and a 2FA request from Digital Ocean

Where 2FA fails is it's adoption rate amongst users: Less than 10% of google accounts utilize 2FA [2]. 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 tool used to combat weak and re-used passwords. Password managers are applications developed that securely store username and password pairs along with the service they are used for so users don't have to either memorize all their credentials or store them in plaintext. 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]. Like 2FA, password managers may be seen to be requiring too much effort in relation to the risk it mitigates.

1.4 Goal

This GP will develop an intrusion detection system (IDS) to support current login systems. 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 used for subsequent login actions. This IDS 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 continue to secure in the event that 2FA is breached as well.

2. Background

2.1 2FA Issues

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 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, may opt for a smoother login experience. Two-factor-authentication is being made 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 [4]. As such, it is important to consider alternative measures of securing login processes that maintains a comfortable user experience.

2.2 Data Modeling

Many Intrusion Detection Systems are developed to target either network layer attacks or monitor command line inputs. 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 such as mouse activity. This adds another factor that may be used when monitoring for malicious activity. Depending on the implementation, this data model may be linked to a session if the system is a shared environment, or an individual account. [5]

2.3 reCAPTCHA

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. reCAPTCHA has gone through three major iterations, each aiming to improve upon its three tenants.

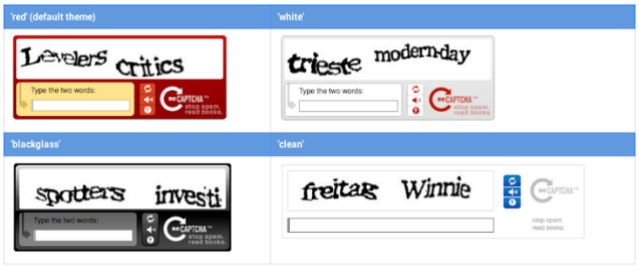

Fig. 3 v1 demonstrating a text distortion challenge

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.

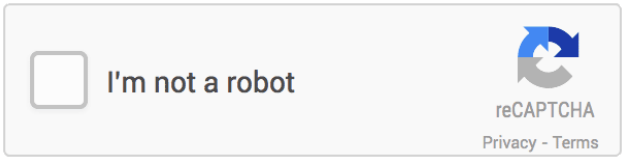

Fig 4. v2 collecting data before deciding to display a challenge

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.

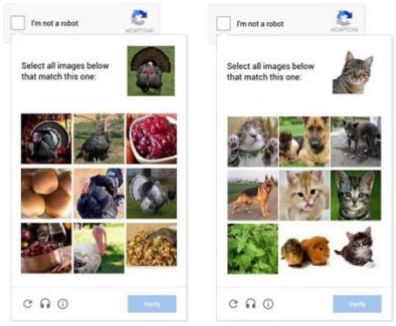

Fig. 4 v2 displaying a challenge

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. By mimicking reCAPTCHA's data gathering techniques, this undergraduate resume attempts to differentiate when the owner of an account logs in to a web page, and when an unauthorized human user attempts to log in with the same credentials. [6]

3. Design

An intrusion detection system is comprised of a frontend that is monitored, and a backend that handles the processing of any data that is gathered. The frontend for this IDS is a web page that users may login to using a username password pair. This web page will serve as the platform on which tests will be conducted and demonstrates how the IDS operates.

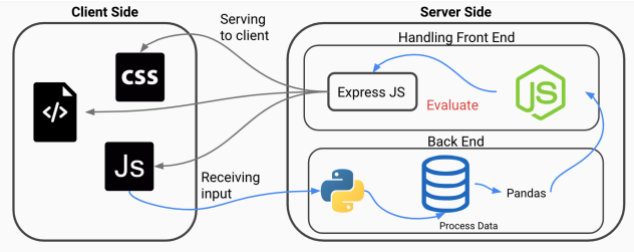

Fig. 5 Overall flow of IDS

3.1 Frontend

NodeJS [7] 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 [8].

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 each user's individual profile page. The profile page will only be accessible once a user has logged in with the appropriate credentials. The login page will contain a form that sends a request to the backend for validation. If the credentials are validated, a further 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.

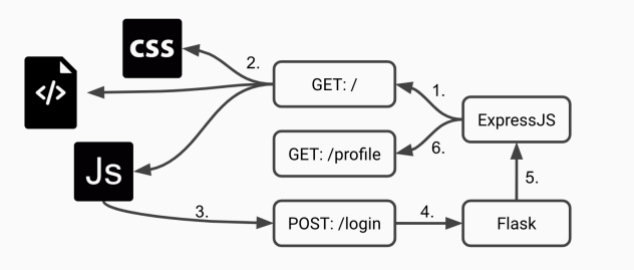

Fig. 6 Flow of frontend

Figure 6 demonstrates the flow of the frontend in 6 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. This route is handled by Flask [9]. Once the backend finishes processing the data, Flask sends a redirect to the client making the browser redirect to the '/profile' page being served by ExpressJS.

3.2 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 database that stores its data objects in a JSON-like format [10]. 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.

In order to receive data from the frontend 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 analyze the data. The Pandas module will be used to analyze the data. Once data has been obtained from both the user input through ExpressJS, and the historical data from MongoDB. Pandas will analyze the data and report back whether or not to authenticate the user.

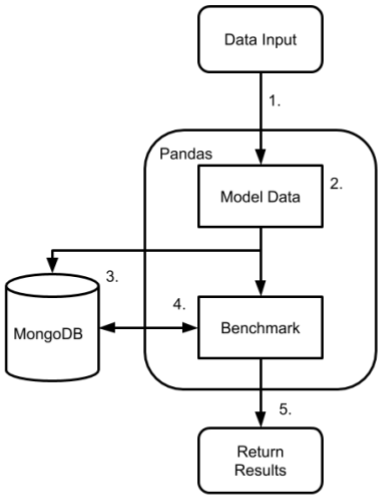

Fig. 7 Flow of backend

Figure 7 details the flow of the backend in 5 steps. The backend first receives login attempt details from the frontend as data input. Using the Pandas library, a model is built for the data. A query is then made to MongoDB for the historic data of the credentials attempting to log in. MongoDB returns this data, which is then used to compare against 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.

3.3 Generating User Credentials

The IDS 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 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 username and password details. Instead, accounts will be prepared ahead of time and assigned to each user.

Random words will be concatenated to form username and password pairs. This ensures that test users will have unfamiliar usernames and 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 combinations of individual characters, numbers, and symbols.

No API was found that provides an interface for generating random dictionary words. Instead it was found that many sites generate their own through internal or opensource datasets. To ensure users are able to type their credentials without

struggles, a set of the 3000 most common English words will be used [11].

3.4 Testing

This undergraduate resume will be evaluated based on how accurate the IDS 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.

Volunteer users will be gathered to provide the IDS 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 quit testing at any time. They will be notified regarding all types of data being gathered during the tests.

The data gathered from the users will be mouse cursor activity, typing activity, browser type, and device type. Mouse cursor activity encompasses both mouse speed and the amount of clicks. On mobile devices, data such as location, accelerometer, and biometrics can be gathered. To preserve the security of users however, these will not be considered for the scope of this study [12]. Mouse movement will also be ignored. 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 IDS 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. During this testing phase, to ensure that false positives or successful logins do not contaminate the existing dataset, a separate API endpoint will be used. The measurements of these attempts will then be saved to a separate collection.

4. Implementation

This resume proposes a method of distinguishing between different human users by monitoring their log in behaviors with the goal of developing an effective IDS. To distinguish between the account owner and unauthorized login attempts, a benchmark 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.

4.1 Gathering User Data

Mouse activity will be defined as the speed of the mouse cursor and the amount of clicks it performs across a login session. 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.

Typing speed is a common type of data gathered for IDS 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. Measuring typing speed by word count is ineffective. This study will instead measure by characters per minute (CPM).

Traditional IDS' 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. This means for each individual user, there is only one expected input. The restriction on input provides an opportunity to be stricter with the measurements. Alongside measuring typing speed in CPM, this study will also measure the delta time between each individual keystroke. Each user has a password of fixed length. Measuring a user's more minute idiosyncrasies may offer more accurate results when distinguishing between individuals. In the following equation, time is represented by t.

$$\overline{\Delta t} = \frac{\sum_1^n \Delta t}{n}$$

Browser and device data will be gathered using the FingerprintingJS2 [13]. This data will be used to separate datasets for users that operate on 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.

4.2 Developing Benchmarks

To develop benchmarks with which to compare user data against, a series of tests will be conducted to measure human error across the various data types. The two main types of data to be examined will be typing activity and mouse activity. By examining the variance within the data users submit over numerous login attempts, a value for human error can be calculated.

4.3 Schedule

This resume serves as an introduction to the study. At the time of writing, the Frontend and Backend are largely completed. Following the submission of this resume, data collection will

begin. The author must first gather volunteers to provide the initial data to measure human error. They will then be asked to undergo the testing process over the course of multiple weeks. During this time, measurements made will allow the author to continually adjust their algorithms and benchmarks to improve accuracy. An undergraduate thesis will be written to document this process and the results. Completion of both these tasks should be within the timeframe of late-June to mid-July.

5. Limitations

This resume goes over the motivation and design behind an undergraduate thesis. During this planning and development phase however, multiple limitations were found that must be addressed.

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 on the client's side to be successfully used. This has led to low adoption rates. Conversely, services such as Google's reCaptcha are providing a method to prevent unauthorized automated activity. Targeting users that forgo these tools and services however means that the IDS 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.

Furthermore, 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 due to the current design of the IDS focusing on a fixed-length credential, the dataset for developing benchmarks will need to be generated once more using the new password, thus lowering the security able to be offered.

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, because the delta times need to be analyzed, they must remain in an un-hashed form. Should the database be leaked, this could give attackers information about the lengths of each individual user's password.

Reference

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

[3] 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.

[4] 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.

[5] 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.

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

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

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

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

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

[11] 3000 most common words in English | Learn English. (n.d.). Retrieved from https://www.ef.com/wwen/english-resources/english-vocabulary/top-3000-words/

[12] 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.

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