Wedge - Textual and Graphical Based Password Authentication

Jeanie Chen and Kiana Hosaka

Abstract
Passwords are an essential form of authentication for users to verify their identities. Text based passwords are the most commonly used method due to the efficient authentication process. However, strong text based passwords are difficult to remember and are vulnerable to shoulder surfing, video recording, and keystroke logging attacks. Graphical based passwords are generally easier for users to remember but can be expensive for computers to process and are susceptible to the same attacks as textual passwords. Therefore, the problem is that both textual and graphical passwords are vulnerable to various kinds of attacks. We propose a solution that combines both textual and graphical based passwords for security against shoulder surfing, video recording, and keystroke logging. To implement this secure system, we conducted research on various existing password authentication schemes. Our authentication scheme, *Wedge*, is inspired by Nevon Projects' scheme which combines both textual and graphical based passwords. However, Nevon Projects' system is not secure from shoulder surfing, video recording, and keystroke logging attacks. *Wedge* is implemented with additional features to provide protection from these attacks, as reflected in our data of participants who tested the ease of use and the security of our textual and graphical based authentication scheme.

Keywords

- **Authentication:** the process of confirming a user's identity
- **Brute Force:** an attempt to crack a password using trial and error, trying as many things as possible hoping to eventually guess correctly
- **Dictionary Attack:** a brute force approach of trying all possible dictionary words to try to find the correct password
- **Graphical Password:** authentication system that allows users to interact with images in various ways such as being selected in a specific order with a graphical user interface
- **Keystroke Logging:** attack on a user by recording or tracking a user's keystroke on a device
- **Recall Based Systems:** type of graphical password where users perform operations that must be repeated during the authentication process
- **Recognition Based Systems:** type of graphical password where icons are used as passwords to strengthen the user's ability to memorize and identify their passwords
- **Shoulder Surfing:** attack on a user of an electronic device to obtain their personal information by watching their authentication process

1. Introduction

There are various ways in which users can authenticate themselves using passwords. The types of passwords fall into categories of what a user knows, what a user has, and what a user is:

- What you know: Knowledge based authentication. Users must remember their text or graphical based password to authenticate their identities.
- What you have: Token based authentication. Users must bring a form of token to authenticate their identities.
- What you are: Biometric based authentication. Users must use some form of their unique biometric properties to authenticate their identities.

Text and graphical based passwords fall under the category of passwords that users know. Currently, text based passwords are more commonly used than graphical based passwords due to the efficient login process, assuming that users remember their passwords. However, text based passwords can be difficult for users to remember because there are various rules that determine the strength of a password. Moreover, passwords are meant to be secure ways for people to gain authorization and access, but there are many ways in which text based passwords get cracked. Text based passwords are vulnerable to attacks such as shoulder surfing, keystroke logging, and video recordings.

An alternative to text based passwords are graphical passwords. While graphical passwords are easier for users to remember, they pose disadvantages for both the computer and user. Various forms of graphical passwords are also still vulnerable to attacks such as shoulder surfing, keystroke logging, and video recording and thus are not adequate authentication processes.

Therefore, the main problem is that both textual and graphical passwords have various disadvantages and are vulnerable to keystroke logging, video recordings, or shoulder surfing. It is an interesting problem because not a lot of companies and systems currently take into account how unsecure passwords are, and the process of simply typing in text based passwords are still widely used to gain access to sensitive and confidential information. Here, we have researched various graphical and textual password models that currently exist and build on how these systems can be improved. We propose a solution for a login system that combines both graphical and textual passwords to ensure a secure and efficient login process that prevents shoulder surfing, keystroke logging, or video recording attacks, and is easy for the user to remember. This report will examine and discuss the advantages and disadvantages of existing authentication methods, and introduce our proposed implementation of a secure authentication system.

2. Related Work

To illustrate the importance of password security and why we propose an improved authentication system, we did much research on the security of textual passwords. In the paper "Weak Password Security: An Empirical Study" by Weber et al., the authors state that passwords are a necessity in our daily lives but "passwords often remain below the level of active, conscious thought for many" (Weber, 1). This means that people don’t usually give much thought into making their passwords more secure. In fact, the SysAdmin, Audit, Network, Security (SANS) Institute indicates that "weak or nonexistent passwords are among the top 10 computer vulnerabilities in homes and businesses" (Weber, 1), which shows the dire importance of addressing the problem of password security. According to Weber et al., the strength of a password is directly related to the length of a password and the number of possible characters that must be examined. However, it is also important to note that length doesn’t always guarantee a stronger password as it also depends on the method used to choose the password. For example, an 8-character password that is an easily guessed dictionary word is vastly less secure than an 8-character long password that contains random characters. While a dictionary attack must search through all possible words listed in a dictionary, it can still be a lot less computationally intensive than a brute force approach, which must consider all possible combinations of characters of a given length. Moreover, attacks can be made even easier with the use of supercomputers. Weber et al.'s article mentions various methods that have been developed to help users create hard-to-crack passwords. A "password phrase", which utilizes a combination of letters and numbers to determine a phrase, is one of the methods stated by the authors. For example, the password “good night to you” could be translated into the password “goodNight2U” which incorporates letters, letter cases, and numbers. These mnemonic passwords have been shown to upgrade the security of the original password. Other secure methods such as two-factor authentication and biometrics have also been

proposed to increase the security of the authentication process. While these methods do improve security, "their cost and complexity of implementation remain barriers to adoption" (Weber, 1). They simply use more resources than traditional textual passwords to implement and are much harder to make universal. Until these tradeoffs are decreased to an acceptable point or eliminated completely, text based passwords will remain the most popular form of authentication.

Although strong text based passwords are less vulnerable to dictionary or brute force attacks, they can be difficult for users to remember and remain vulnerable to more sophisticated attacks such as shoulder surfing, video recording, and keystroke logging. According to the article "Shoulder Surfing Proof Graphical Password Authentication Scheme" by Wu et al., obtaining a strong and easily memorable textual password is difficult. While the "password phrase" method mentioned above is an effective step towards password strength, Wu et al. states that there are various other rules to follow to achieve a strong textual password. These rules go as follows: password length must be long, words must be original, storage location of passwords must be secure, passwords must be changed periodically, system generated passwords are encouraged, and different passwords must be used for different systems. If users closely follow these rules to obtain a strong password, they are more likely to have a password that is difficult to remember. Wu et al. also describes how even with a strong textual password, passwords will not be able to withstand shoulder surfing attacks. The authors state that "when a user enters the password in a public space, it will be compromised if there is any peeper behind" (Wu et. al, 246). Video recording is a more advanced attack that involves attackers using video equipment to capture the user's password. This recording could capture the keys that the user pressed as well as the locations of the mouse clicks. Moreover, Amos P. Waterland explains another attack, keystroke logging, in his article "Secure Password Entry". In a keystroke logging attack, a malicious keystroke logging device can be planted into the hardware of a keyboard. This device has the capability to record and timestamp the keystrokes that have been typed on the keyboard. After recovering the device, the attackers have the ability to search the log for passwords used during authentication. Waterland states that this is typically an easy process because "the string of characters typed after an authentication challenge to which the user responds is the password" (Waterland, 1). Strong text based passwords provide users with protection from dictionary and brute force attacks, but can be easily forgotten and cracked through more advanced attacks such as shoulder surfing, video recording, and keystroke logging.

In an attempt to discover more secure password authentication systems, we researched the security of graphical passwords. Author Paulen wrote in "Taking a Graphical Approach to the Passwords" that "graphical approaches are more intuitive and offer more potential passwords than text-based systems" (Paulsen, 19). Therefore, graphical passwords are easier for users to remember and are more difficult for attackers to crack. In Wu et al.'s article, the authors determined that there have been various previous attempts of graphical passwords. According to the authors, there are two categories of graphical password systems: recognition based systems and recall based systems. Recognition based systems use icons as passwords to strengthen the user's ability to memorize and identify their passwords, while recall based systems have users perform operations that must be repeated during the authentication process. Firstly, Wu et al. explored the various related work of recognition based systems. These systems included that of Dhamiha Perrig, who created a system that randomly generated various icons onto a screen and had users select them in a predetermined sequence during the authentication phase. Komanduri Hutchings proposed a system where users selected icons as passwords, had the system randomly display various icons, and had users select the correct icons that they had chosen as their password. Next, Wu et al. explored the various related works of recall based systems. Wiendenbeck et al. created a system where users chose a graph and successive points on the graph. During authentication, users clicked on the points that they had preselected in a sequence. Jermyn et al.'s system had users write words or drawings as their passwords.

Users had to then draw the same words and drawings to be verified. Malek et al.'s proposed system had users draw an image on a touchpad with a desired pressure, and had them repeat the exact process to authenticate themselves. In Suo et al.'s system, users selected icons from a database as passwords and determined a location to place them on the screen. The users then placed them on the same screen locations during authentication.

While these proposed systems were creative and had many strengths, each system had faults. For example, in Malek et al.'s system, users struggled to authenticate themselves because they were unable to replicate the exact pressure that they had used to register their password. Wu et al. concluded that recognition based systems reduce system efficiency due to the vast number of icons that the system must process. In recall based systems, users may have difficulty repeating the exact same steps as the operations selected during registration. In addition to the cons stated, some of the current graphical passwords are still vulnerable to keystroke logging, video recording, and shoulder surfing.

The most interesting and secure password scheme that we found came from Nevon Projects, a website that provides users with various mechanical and software engineering project ideas. Nevon Projects proposed a combined text and graphical based password scheme that they claimed to be resistant to shoulder surfing. In this authentication system, users create a password during the registration phase based on a given range of letters and numbers. The user also selects a color from a given range. During the login phase, the system displays a wheel composed of colors, numbers, and letters. The numbers and letters are displayed in either an inner orbit and an outer orbit. The colors in the wheel are rotated by pressing the "clockwise" or "anticlockwise" buttons displayed on the screen. While the colors rotate, the characters remain stationary. To enter their password, users rotate the wheel until their selected color during registration is on a slice that contains the first letter or number of their password. The user then presses the "inner orbit" or "outer orbit" button displayed on the screen depending on the position of their first letter or number. The user will continue to follow these steps until their entire password has been entered.

3. Main Body of Work (Design and Implementation)

While Nevon Projects' approach certainly improves upon the previously mentioned authentication systems, there are various vulnerabilities remaining in the system. First, the system claims to be immune to shoulder surfing, but having the user press "clockwise", "counterclockwise", "inner orbit", and "outer orbit" buttons on the screen is not entirely safe from shoulder surfing because attackers could easily see or record the password being entered. Nevon Projects' authentication system also does not implement the randomization of the letters, numbers, and colors that appear on the wheel when a user opens the login screen. While an attacker may not know a user's password or color, the attacker could video record the login process and successfully login as the user by selecting the exact same buttons as shown in the recording. If the position of the letters and numbers are consistent during every login attempt, the system is also vulnerable to keystroke logging. Moreover, the process of using a mouse and a button on the graphical interface to rotate the wheel and select letters or numbers increases the login time significantly, making the system more vulnerable to shoulder surfing attacks.

We propose an improved and more secure authentication system called *Wedge* in the form of a Python application and utilizing Python's graphical user interface library, Tkinter. In Nevon Projects' system, the user pressed buttons displayed on the interface to rotate the wheel and select numbers or letters. Our system uses the keyboard characters "w", "a", "s", "d" to provide for a more discrete login process. This also provides users with a faster login than Nevon Projects' system by removing the use of a mouse,

thereby decreasing the risk of successful shoulder surfing attacks. *Wedge* integrates more colors, letters, and numbers to display on our system to increase security and freedom of passwords selected during registration. Furthermore, *Wedge* randomizes the locations of the letters, numbers, and colors on the wheel for every application launch. Because of these additional features, attackers will have difficulty tracking keystrokes, shoulder surfing, or analyzing a video recording to crack the user's password. Below is a description of how users can interact with our software:

To launch our application, the user can download the *Wedge* repository from GitHub, then navigate into that directory and run the command "python3 main.py" in terminal. After the user launches the application, a pop-up home window appears that gives the user two options: Log In or Register. The user can choose to register by providing an original username, a textual password, and a color chosen from a drop down menu. It is critical that the user remember these three pieces of information. The user can then go to the login page where they will enter their username and press "enter". Afterwards, the user will use the keyboard letters "w", "a", "s", "d" to simulate the direction keys to turn the color wheel. "a" corresponds to turning the colors in the wheel counterclockwise and "d" corresponds to turning the colors in the wheel clockwise. When the user's selected color lands on the first character of their password, they will select "s" or "w". "s" corresponds to selecting the letter or number in the inner orbit, and "w" corresponds to selecting the letter or number in the outer orbit. Using these keys, users will repeat this process, choosing the appropriate letters and numbers until their password is built. Once the password is entered, the "Login" button can be clicked and another pop-up window will appear, informing the user of either login success or failure.

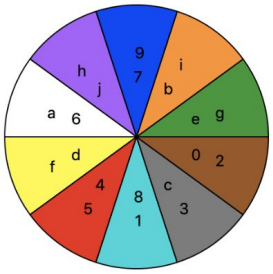

Fig. 1. *Wedge* Password Authentication System

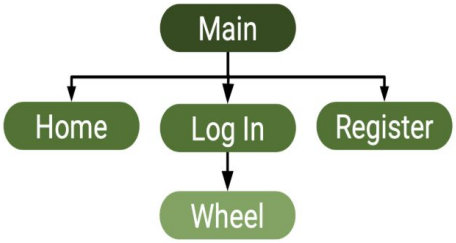

Fig 2. Software Architecture of *Wedge*

Because Python is an object oriented programming language, we designed the architecture to have different components and classes that interact with each other. The Main class is in control of routing between the three different modules and windows: Home, Log In, and Register. Each of these "windows" is also its own class with its own appropriate functions that contain the actions to be performed on that page. For example, the Register module is where the user creates an account and selects a username, password, and color. This module also stores each user's credentials to a text file and ensures that no duplicate usernames exist. The Log In module contains functions to process keyboard inputs from the user as they interact with the wheel to build their password. It also contains functionality for authentication verification when the user clicks the "Login" button after entering the password.

```
# function that handles keyboard input
def keyPress(self, event, wheel, canvas, password_canvas):
    key = event.char
    # print(key, 'is pressed')

    # if key pressed is w (up), user chooses outer letter
    if key == "w":
        index = 1

        # need to show a * on the screen, find the slice that is chosen,
        # get chosen char from slice,
        # add char to current password
        slice = wheel.getSlice(self.user_color)
        char = slice.getCharacter((index))
        self.current_password += char
        password_canvas.create_text(DISPLAY_PASSWORD_COORDS[0], DISPLAY_PASSWORD_COORDS[1], text="*")
        DISPLAY_PASSWORD_COORDS[0] += 10
```

Fig. 3. Snippet of keypress handling for when user selects "w" key

The main challenge for this project was determining how to create the color wheel, make it interactive, and integrate all the functionalities we wanted for our system. We decided to create two classes for the wheel: Slice class and Wheel class. The Slice class simulates each slice of the wheel and contains a list of the two characters that will get drawn on the slice. It also contains the functions that draw the characters onto the interface.

```
# a slice has 2 characters in list form and it's start position
class Slice:
    def __init__(self):
        # characters is list where index 0 represents the inner letter
        # index 1 represents the outer letter
        self.characters = []
        self.start_angle = None
        self.position = None
```

Fig. 4. Snippet of Slice class

The Wheel class simulates the entire wheel, and it contains a list of the slices, predefined colors and characters to assign to each slice, and a dictionary that keeps track of which color is associated with which slice at any moment. This class also contains functionality that calls the draw functions for each slice and updates the wheel after each rotation. Most of the essential functionalities of our secure login system exist in the Wheel class.

```
# a class that contains a list of colors, password characters, and Slice classes
# manages all the slices and takes care of wheel big picture stuff
class Wheel:
    def __init__(self):
        self.colors = ["red2", "DarkOrange1", "blue", "forest green", "purple1", "white", "saddle brown
                       , "dark turquoise", "yellow", "dim gray"]
        self.slices = []
        self.characters = ["1", "2", "3", "4", "5", "6", "7", "8", "9", "0"
                           , "a", "b", "c", "d", "e", "f", "g", "h", "i", "j"]
        # ordered dictionary to keep track of which color is at which slice
        self.pairings = OrderedDict()
        # could possibly change number of slices
        # used to calculate the extent to draw the arcs
        self.number_of_slices = 10

        # populate slices list
        self.makeSlices()
```

Fig. 5. Snippet of Wheel class

4. Results and Analysis

The result of the research and implementation of *Wedge* is a secure textual and graphical based password authentication scheme that provides users with protection from shoulder surfing, keystroke logging, and video recordings. *Wedge* only requires users to remember a username, a simple textual password, and a color, which reduces the difficulty for users to remember their authentication information. While our system does not fall directly into either category of being classified as a recognition based graphical system or a recall based graphical system, it falls more closely under a recall based system. In a recall based system, users replicate the exact same actions as executed during the registration phase. *Wedge* might be considered a recall based system in that users will replicate the same process, but the steps won't be the exact same every time because of the system's randomization functionality.

To test the security of our system, we asked 21 participants to act as attackers and attempt to determine our passwords to the system. We had participants carefully watch us log into *Wedge.* Moreover, we instructed users to video record our login process. After watching us login as well as reviewing the video footage they took, we asked users to report their guess of our passwords. Only one of the participants, Participant #21, was able to determine the password, let alone understand the authentication scheme. Participant #21 spent significantly more time analyzing the video footage than other participants and relied heavily on watching which keys were pressed. This could have been made harder for Participant #21 to crack if we had covered our hand that was navigating the "a", "s", "d", "w" keys to prevent them from seeing what keys were pressed. Because Participant #21 was the last participant that we asked to act as an attacker, we were unable to use the hand-covering technique on previous participants. However, Participant #21 stated that if we had covered our left hand while logging into the system, he would have been unable to determine the password. Upon explaining to the other participants how the scheme worked, they were still unable to determine our passwords.

With a password of length n and our system supporting 10 colors and 20 characters, there are a total of 10 * 20^n (let’s call this number X) combinations of passwords and colors that a user can create. This means that a user’s password is 1 out of X possible passwords. To put it into perspective, for a password of length 5, attackers would have to attempt 32000000 passwords in a worst case scenario. Based on our sample of 21 people, this probability supports the data that our system is 95.24% secure because it is difficult for attackers to determine a password by shoulder surfing and video recording without computing power.

To test our system's ease of use, we asked the same 21 participants to act as users of the system. *Wedge*'s ease of use is determined by the speed in which users are able to log into the system. This is important because a quick authentication attempt provides another layer of protection from shoulder surfing. For consistency, we had participants register a username of length five as well as a password of length five. Specifically, users were instructed to register their username as "kiana", their password as "hi000", and any one of the provided colors. After allowing participants to register their credentials, we timed how long it took for participants to log into our system with two attempts. Because our system does not support deleting characters after they have been selected, 4 participants had to start the authentication process over after accidentally selecting an incorrect character during their first attempt. After the first attempt, the 21 participants were able to enter their username and password within a range of 17.92 and 29.1 seconds, with an average of 21.367 seconds. After the second attempt, participants were able to enter their username and password within a range of 11.81 and 23.76 seconds, with an average of 17.937 seconds.

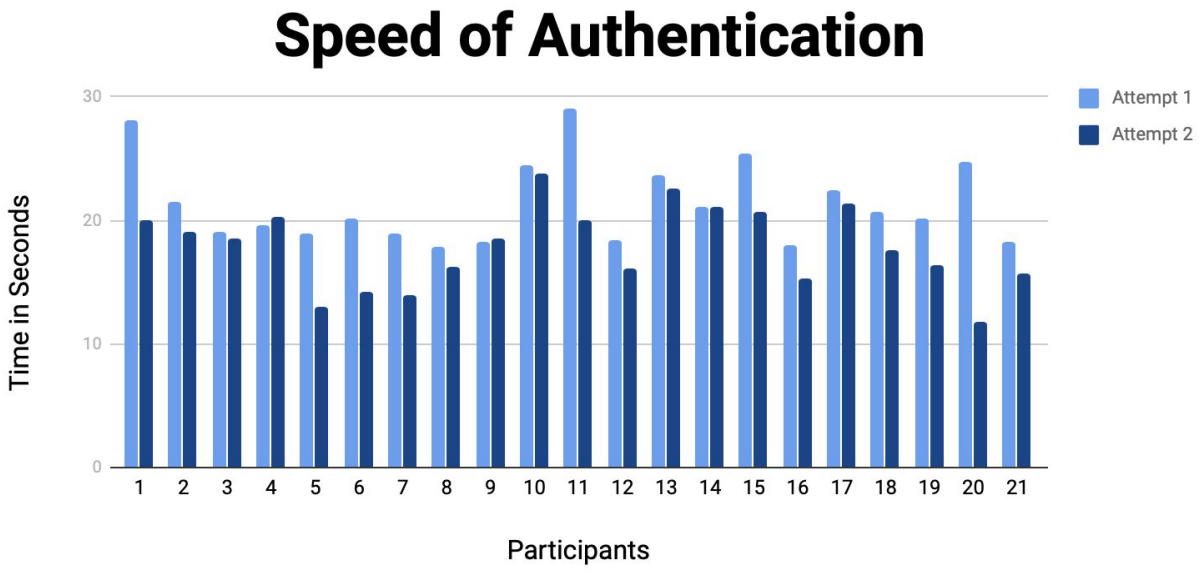

Fig. 6. Chart indicating speed that users were able to log into the system.
Participants on x-axis and time in seconds on the y-axis.
Light blue indicates attempt 1 and dark blue indicates attempt 2.

Through our data, we have concluded that participants generally improved their speed upon their second attempt of authentication. The two person *Wedge* development team has been able to log into the system with a speed of 6.99 and 10.41 seconds because we have had practice using the system throughout the development process. From our personal experience as well as the data provided above, we believe that upon several attempts of logging in, users would be able to improve the speed in which they could authenticate themselves using *Wedge*.

From our 21 participants, we asked for any feedback that they were willing to provide us. Participant #21, who was the only participant that determined the password, suggested that we implement randomization in our system after every character is selected rather than solely upon each system startup. This suggestion was a functionality that the *Wedge* development team had considered, but had decided not to implement because it would increase system complexity and user login time, therefore decreasing ease of use. We also received feedback from our participants who agreed that because they were unfamiliar with the usage of the "a", "w", "s", "d" keys to navigate the wheel, they spent more time during the authentication process. They claimed that with practice, they would be able to quickly authenticate themselves as the *Wedge* development team is able to. Participant #2 suggested that we display a legend for what the "a", "w", "s", and "d" keys do, making it easier for users to remember how to enter their password. Again, the *Wedge* development team had considered this functionality during the design process, but determined that this would decrease the security of our system by displaying to attackers the keys to navigate our system. As mentioned previously, we also predict that with time, users will be able to increase their familiarity with the keys and therefore decrease their login time.

While we were unable to directly test our system's security from keystroke logging, we believe that the system is protected from keystroke logging as well. Because the colors and characters are randomized on the wheel, a keystroke logging attack would fail because the keystroke log would reveal different keys after every authentication challenge.

Wedge has been implemented to completion in terms of usability. However, there were some improvements that we wanted to implement in our system. We would have liked to include the full range of the alphabet as well as upper and lower case letters but concluded that this addition would require more wedges to be added into the wheel, increasing complexity and decreasing readability. These additional characters would also increase the time it takes users to log into *Wedge.* Another feature that we would have liked to implement is to enable users to undo what they have chosen on the wheel using the "delete" key on the keyboard. In our system, the user must restart the login process if they choose the wrong character. While our system is protected from keystroke logging, video recording, and shoulder surfing, we did not implement security for the credentials file. The system stores the user credentials in a flat text file that does not offer further security insurances such as hashing or encryption. We also acknowledge the limitation of speed during the authentication process. *Wedge* requires users to carefully rotate the wheel and determine the location of the letters and numbers on the inner or outer orbit, which takes more time than commonly used text based passwords. However, as shown in our data above, participants were generally able to improve their speed even after their second attempt of using the system and vocalized their expectation of improvement in speed over time.

5. Discussion of Issues

While the *Wedge* development team did not encounter large obstacles, we faced minor challenges. Firstly, we had to teach ourselves the Python GUI, Tkinter. While this was not a particularly difficult task, we found that it was a tedious process that took more time than we anticipated. Another issue that we faced was placing the letters and numbers on the wedges of the wheel. While we assume that an efficient formula to determine the placement of the letters and numbers with ease exists, we had difficulty implementing such a formula. Rather, we used a brute force approach and manually placed each letter and number on the wedges of the wheel. In regards to implementation, the main challenge we faced in the beginning was determining how to implement an interactive wheel that had all the functionalities we wanted. We eventually decided on creating the wheel with 2 classes, the Slice class containing Tkinter functionalities and the Wheel class containing the functionalities of the wheel.

Moreover, the implementation of our system was more time consuming than we had projected it would be. As mentioned in 4. "Results and Analysis", there were many improvements that we had wanted to integrate into our system. While we had begun implementing a few of those functionalities, we ended up having to disregard them entirely due to lack of time.

6. Conclusions and Lessons Learned

In conclusion, a problem in today's commonly used text based password authentication schemes is that the systems are vulnerable to various attacks. Text based passwords are generally not secure from brute force, dictionary, keystroke logging, shoulder surfing, and video recording attacks. Moreover, strong textual passwords are difficult for users to remember. Some graphical based passwords are still vulnerable to the same attacks. Recognition and recall based graphical systems both have vulnerabilities, where recognition based systems are expensive for computers to process and recall based systems require steps that can be difficult for users to exactly replicate during authentication. While various graphical based password authentication schemes have been proposed to increase the security of the authentication process, these proposals have vulnerabilities or are challenging for users to interact with. Therefore, the proposed system, *Wedge*, was created to provide a simple interface and protection from shoulder surfing, video recording, and keystroke logging by combining text and graphical based authentication techniques.

This system requires a user to remember a textual username, textual password, and a color. With this information, the user is able to navigate the graphical interface to enter their passwords. According to 21 participants, the system provides for a 95.24% security rate, and takes an average of approximately 17.937 seconds to log into the system on a second attempt.

Throughout the development of *Wedge,* we learned a significant amount through our research of various textual and graphical based authentication schemes and also enjoyed programming these systems. Neither of us had ever considered the vulnerabilities of the text based passwords that we use daily. We find ourselves being more careful when logging into our personal computers as we are now well aware of the ease in which attackers could watch or video record us logging in. We had originally expected to find that graphical based authentication schemes were the obvious solution to the vulnerabilities that arise with text based passwords. We were surprised to find that graphical based passwords have their own vulnerabilities, some even sharing the same vulnerabilities as text based passwords. While our authentication scheme has weaknesses as well, we hope that this approach brings insights to our readers and will be continually developed and improved in the future to eventually provide immunity from shoulder surfing, video recording, and keystroke logging attacks.

7. References

Dhamija, R., Perrig, A.: Deja Vu: a user study using images for authentication. In: Proceedings of the 9th Conference on USENIX Security Symposium, pp. 45–58 (2000)

Jermyn,I.,Mayer,A.,Monrose,F.,Reiter,M.K.,Rubin,A.D.:The design and analysis of graphical passwords, Proceedings of the 8th USENIX Security Symposium. Washington, D.C., USA (1999)

Komanduri, S., Hutchings, D.R.: Order and entropy in picture pass- words. Graph. Interface 322, 115–122 (2008)

Malek, B., Orozco, M., Saddik, A.E.: Novel shoulder-surfing resis- tant haptic-based graphical password. In: Proceedings of the Euro- haptics Conference, Florence, Italy (2006)

Paulson, Ld. "Taking a Graphical Approach to the Password." *Computer* 35.7 (2002): 19. Web.

Suo, X., Zhu, Y., Owen, G.S.: Analysis and design of graphical password techniques. Adv. Visual Comput. 4292, 741–749 (2006)

Waterland, Amos P. “US7539874B2 - Secure Password Entry.” *Google Patents*, Google, patents.google.com/patent/US7539874B2/en.

Wiedenbeck, S., Waters, J., Sobrado, L., Birget, J.C.: Design and evaluation of a shoulder-surfing resistant graphical password scheme. In: Proceedings of the Working Conference on Advanced Visual Interfaces, pp. 177–184 (2006)

Wu, Tzong-Sun, et al. “Shoulder-Surfing-Proof Graphical Password Authentication Scheme.” *International Journal of Information Security*, vol. 13, no. 3, June 2014, pp. 245–254. *EBSCOhost*, doi:10.1007/s10207-013-0216-7.

Weber, JamesE., Guster, Dennis, Safonov, Paul, Schmidt, MarkB. “Weak Password Security: An Empirical Study.” Information Security Journal: A Global Perspective, 19393555, Jan2008, Vol. 17, Issue 1.

“Graphical Password To Avoid Shoulder Surfing.” *Nevon Projects*, 24 Nov. 2018, nevonprojects.com/graphical-password-to-avoid-shoulder-surfing/.

Appendix

System Requirements

- *Wedge* has been tested on macOS Mojave and macOS Catalina using Python3
- *Wedge* is not compatible with Dark Mode

Downloading Repository

1. Go to our Github repo website https://github.com/kahosaka/wedge
2. Click “Clone or Download” and copy the URL
3. Open terminal and navigate into the directory where user wishes to place the repository
4. Run the command “git clone https://github.com/kahosaka/wedge.git”

Running *Wedge* Application

1. Complete the steps in “Downloading Repo”
2. In the terminal and in the Wedge directory, run the command “python3 main.py”
3. A window named “Wedge” will open

Home Page

- Two options that a user can do on the home page: log in or register
- Clicking on the “Register” button will take the user to the Registration page
- Clicking on the “Log In” button will take the user to the Log In page

Registration

1. Once on the Registration page, type in the chosen username in the text box underneath the text “Username”
2. Type in a chosen password using only the characters specified above the password text box
3. Choose a color from the drop down menu
4. Remember username, password, and chosen color in order to log in
5. Submit account information and wait for the pop up window to inform if registration was successful
6. If successful, click the “Log In” button to go to the log in page
7. If unsuccessful, restart steps 1-3

Login

1. Once on the Log In page, type in username in the text box and hit the “enter/return” on the keyboard
2. Find the slice on the wheel that has chosen color
3. Rotate that colors counterclockwise or clockwise using the keys “a” or “d” respectively until color is on a slice that contains the first character of user's password
4. Select the inner or outer character using the keys “s” or “w” respectively
5. Repeat steps 2-4 for every character of the password until password has been completely entered
6. Click the “Login” button to complete the log in process
7. A pop up window will appear informing log in success or failure