Thesis Dissertation

MEASURING USER HABITS IN MANUALLY TYPING PASSWORDS

Katerina Erodotou

UNIVERSITY OF CYPRUS

COMPUTER SCIENCE DEPARTMENT

June 2022

UNIVERSITY OF CYPRUS

COMPUTER SCIENCE DEPARTMENT

Measuring User Habits In Manually Typing Passwords

Katerina Erodotou

Supervisor

Dr. Elias Athanasopoulos

Thesis submitted in partial fulfilment of the requirements for the award of degree of Bachelor in Computer Science at University of Cyprus

June 2022

Acknowledgements

First and foremost, I would like to express my sincere thanks to Dr. Elias Athanasopoulos, my thesis dissertation supervisor. Without his guidance, valuable advice, and positive approach this project could not have been accomplished. Furthermore, I would like to thank my family for their practical and mental support throughout the past four years of my studies. Lastly, I would like to express my gratitude for my friends. Especially, my classmate Chrystalla Anastasiou who has been by my side through both the challenging and enjoyable parts of our studies.

Abstract

Passwords are still the most widely used technique for user authentication. However, it is unclear how frequently individuals type their password to login to services.

To gather information related to this matter, we use two different techniques. Firstly, we conduct a user study regarding authentication methods. The main purpose of the study is to determine how often users type their passwords to login, register or change their password. Furthermore, to validate the results of the study, we develop a browser extension that records the actions mentioned above and information related to them.

The findings of this thesis can yield insights into which authentication methods are mostly used today. In addition, since there are attacks that depend on specific authentication procedures (e.g., password typing), the results can be further utilized to determine which attacks are more likely to occur. Besides that, there are defenses relied on the fact that users authenticate themselves by inserting a password. Consequently, our results can be used to determine how effective a defense can be.

Contents

Chapter 1

Introduction

Recent technological breakthroughs have resulted in a plethora of personal computing devices, including smartphones, tablets, smartwatches, and many others. This has facilitated the development of a world where technology is an integral part of everyday life. Therefore, ensuring that users are protected in the digital world is critical. Undoubtedly, user authentication methods have a key role in doing so. The most frequently used method for authenticating users is text-based passwords. However, utilizing passwords for user authentication is known to be vulnerable to a variety of attacks e.g., dictionary and spyware attacks [32], [17]. As an alternative, different mechanisms have been proposed in the past years such as biometrics, token authentication, single sign-on, CAPTCHAs etc. [2], [10], [19]

In this thesis, we attempt to discover how often the average user enters their password to register, login or update their credentials. Since there are attacks that depend on password typing [16], [22], [23], [24] our findings can be an indicator to which attacks are more frequent today. Additionally, some existing defenses are relied on users authenticating themselves explicitly by entering their password [13], [14], [29] which is a rather strong assumption. Hence, our results can also be used to determine how effective a defense can be.

A very distinctive example of an attack that depends on password typing is the keylogger attack. Keylogging is based on recording the keystrokes on a machine. The intention of this attack is that the individual using the device will be unaware that their activity is being monitored. Thus, the adversary will be able to steal the credentials of the victim and other private data [26]. Another kind of attack that depends on password-entry events is phishing. For instance, fraudulent websites appear to be legitimate and attempt to harvest the victim's personal details on login actions [1].

The use of text-based passwords as a means of authentication has been extensively studied in the field of computer security. On the other hand, the frequency with which users authenticate explicitly with their passwords has received far less attention. One of the possible reasons that this matter has not been evaluated before is that there is a challenging part to it. One way we can approach this matter is by conducting a survey related to authentication methods. Nevertheless, in order to validate the results of the study we should also develop tools that track user behavior and authentication methods, which can be difficult.

The results can be valuable both on a collective and an individual level. Data breaches have affected millions of accounts even for the most prominent companies such as Facebook, LinkedIn, Yahoo and Twitter [3], [4], [7], [12]. However, even for small businesses the average cost of a data breach is $2.35 million [15]. Taking these points into consideration, no one can argue with the fact that cybercrime can cause significant losses on income to a company [18].

One of the most recent studies we came across regarding password-entry events is one published in 2016 which suggests that users enter their password at least once per day [30]. However, prior research of 2007 reports that the average user has 8 to 23 password-entry events every day [6]. These statistics are quite likely to have altered since then. A more in-depth analysis is provided in Chapter 7.

This thesis makes the following contributions:

- We discuss and assess the results of a user study on authentication methods, and we measure how frequently the participants state that they type their password.
- We design, implement, and evaluate a chrome extension to validate the results of the user study.

Chapter 2

Background

Contents

2.1. Overview

In this chapter, we provide background information which we consider significant for understanding the rest of this thesis. More specifically, we describe browser extensions, the Apache Server, and Selenium Testing.

2.2. Browser Extension

Extensions are programs that can be installed into a browser to change its functionality. This can include adding new capabilities to the browser or changing its current behavior [5]. Some examples of the functionalities that can be added to a browser are advertisement blocking, password management e.tc. In general, extensions provide a wide range of extra functionalities, so that a user can personalize their browser and perform tasks easier.

2.3. Apache Server

To implement this thesis, we used an Apache Server. The main functionality of the Apache Server is to serve as a bridge between the server and the client machines. We chose Apache out of many other web servers since it is simple to customize, dependable and secure [8]. To control the behavior of the server we used config files. To be more precise, we have modified these files to define the IP address that Apache listens to. We also did some changes so that we would be able to receive requests from HTTPS clients. Moreover, we should mention that we have deployed the Apache Server on a Linux machine at the University of Cyprus. Therefore, we receive requests only from clients inside the university's network. Lastly, to handle these requests we have created a PHP script.

2.4. Selenium Testing

As the world is moving towards the digital era, software testing is a necessity rather than a requirement. To test our experiment, we used Selenium Testing. More specifically, we used Selenium WebDriver which is a cross platform testing framework [28]. This tool is used for automating testing on web-based applications in order to ensure that they perform as expected. We should also mention that it is not required to install a selenium server as the test scripts interact directly with the browser. Furthermore, Selenium WebDriver provides us with the opportunity to choose a programming language to create test scripts. In our testing, we use JavaScript. Selenium tests can be written in a way that web elements can be identified. Then, we can perform actions on these elements to mimic the behavior of our program and create test cases.

Chapter 3

Architecture

Contents

3.1. Overview

This chapter demonstrates the general design of our implementation without explaining the technical aspects too much. One of the key components of our architecture is a user study which we conduct using a questionnaire. The main purpose of the study is to give us an insight on how often users believe they manually enter their password. To validate the results of the questionnaire, we also created a browser extension that counts how many times a user manually types their password. Lastly, we use an Apache Server to store our data and a python program to analyze them.

3.2. Google Chrome Extension and Apache Server

Extensions are programs that can be installed to a browser and add new functionalities to it. For our experiment, we created an extension that collects information when a user logs in, registers, or changes their password. As we can see in Figure 3.1 each time one of these actions is executed, the extension sends a request with the information to the Apache Server.

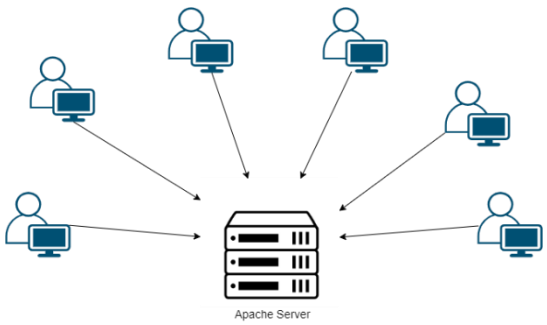

Figure 3.1: Client and Apache Server

We configure the Apache Server so that it accepts POST requests. More specifically, the requests are sent to a PHP file on the server which is responsible for storing the data to a text file. Note that the information we keep about each action is the email of the user (hashed), the website, the date, and a description of the action. The action could be one of the following LoginTyped, LoginNotTyped, Register, or ChangePassword.

To understand the functionality of the extension, we can see the Figure 3.2. To begin with, once the extension is installed, we follow a procedure to get the email of the user. In the case that the user is synced in, the Background Script gets the email of the user from the browser storage. However, if the user is not using synchronization, to get the email of the user, we use a Popup Script. The user clicks on the popup, enters their email, and then clicks the submit button. Once this process is completed, the email is stored in the local storage of the browser.

Furthermore, we process the DOM of each website the user visits by utilizing the Content Script. To be more specific, we attempt to identify the action of password typing. Each time this action is executed, the extension collects the needed information and sends it to the Apache Server.

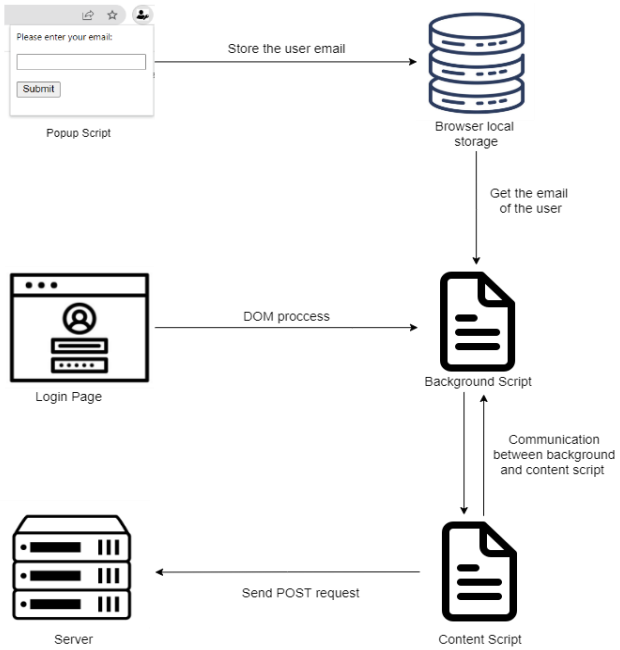

Figure 3.2: Extension

3.3. Python Program

To analyze the data, we created a python program. We study the collected information with respect to two different factors. Firstly, we want to explore how many password-entry events a user has in a day. Therefore, we calculate the duration of the experiment, the number of participants and the total number of requests we received. Then, we estimate the average of how many passwords a participant enters in a day. Secondly, we want to investigate if the user behavior varies depending on the type of the website. For example, some users might type their password more frequently on websites related to their work than they do on social media. Subsequently, we divide the requests into website categories and calculate the total number of requests for each category.

3.4. User Study

To investigate how users log in to their accounts, we also conducted a questionnaire that collected basic information about the user and how they tend to login to their accounts. The questionnaire was answered by 40 participants. We tried to find participants from different backgrounds and age groups so that we could include as much diversity as possible. Our questionnaire contained a set of questions related to how users tend to login to their accounts when using different devices such as smartphones, computers and devices that do not belong to them. We were also interested in the use of password managers, cookies, single sign-on e.tc. Lastly, we asked the participants questions related to how frequently they update their password and possible reasons to change it.

Chapter 4

Implementation

Contents

4.1. Overview

This section is focused on the technical aspects involved. The implementation is based on three main components. First and foremost, we have created a chrome extension which collects information when a user logs in, registers, or changes their password on a website. Whenever one of those actions is completed, the extension sends a request with the information to the Apache Server. Finally, the data is stored in a file on the Apache Server, and further analyzed using a python script.

4.2. Google Chrome Extension and Apache Server

We implement two different methods to capture the action of login, register and change password. The first approach is based on recognizing the action of typing in a password field and clicking the submit button. Therefore, we add an event listener to the document (DOM) which is triggered when any key is pressed, and it executes the *logKey* function.

```
1   document.addEventListener('keydown', logKey)
```

Figure 4.1: Capture Keydown events

The main functionality of *logKey* is to check if the user has typed in a password field and set the variable *typed* to true.

```
function logKey(e) {
  let type = ""
  if (e.target != null && e.target.type != null)
      type = e.target.type.toLowerCase()
  if (e instanceof KeyboardEvent && type.includes("password")
    && typed === false) {
        typed = true
        clickedSubmit = false
   }
}
```

Figure 4.2: logKey function

For the action to be completed, besides typing in a password field, the user must also click a submit button.

Thus, we also add an event listener for click events. Whenever the user clicks an element of the document, the event is triggered and the function *logPasswordSubmit* is called. This function checks if the element is a submit button and then it uses the functions *isLoginElement, isRegisterElement, isChangePasswordElement* to determine the action of the button.

Then, based on the action of the button and the value of the variable *typed*, we define the general action. As we can see in the Figure 4.3 below, the action can be:

1. LoginTyped
2. LoginNotTyped
3. Register
4. ChangePassword

In the case of Register and Change Password, there is only the option *Typed* as a user cannot change their password or register without typing their password.
Before calling the *logAction*() function, we reinitialize the variables *clickedSubmit* and *typed* so that we will capture the next action correctly.

```
If (e instanceof MouseEvent && clickedSubmit === false && action === "")

    // check if the element is a submit button
    if (classname === "submit" || type === "submit" || type === "button" ||
        parent_type === "button" || parent_type === "submit") {

        // check if the user has typed their password
        if (typed === true) {
          if (isLoginElement(target_text) === true || text_sibling === "next"
             || text_sibling === "επόμενο" || text_sibling === "επομενο")
               action = "LoginTyped"
            } else if (isRegisterElement(target_text) === true) {
               action = "Register"
            } else if (isChangePasswordElement(target_text) === true)
               action = "ChangePassword"
        } else if (typed === false) {
          if (isLoginElement(target_text) === true || text_sibling === "next"
              || text_sibling === "επόμενο" || text_sibling === "επομενο")
              action = "LoginNotTyped"
        }

        // reinitialize the variables & call logAction (send data server)
        if (action !== "") {
            clickedSubmit = true
            typed = false
            logAction()
        }
}
```

Figure 4.3: logPasswordSubmit function

The second method we use to identify the *login*, *register*, or *change password* actions is by using the forms of a document. Whenever a form is submitted, we check whether the form concerns one of the actions we mentioned above and if so, we send the request to the Apache Server.

To begin with, once the document is ready, we find all the forms of the document and store them in a variable.

```
Let forms = document.forms
```

Figure 4.4: Get the forms of the document

Then, we analyze the forms one by one and based on their elements we check if the current form is a login, a register, or a change password form. If any of these forms is found, we add to it a submit event listener which executes *logSubmitLogin, logSubmitRegister, logSubmitChangePassword* when it is triggered. These functions are used to send the request to the Apache Server.

```
// find all forms of the document
let forms = document.forms
let elements

// for each form
for (let i = 0; i < forms.length; i++) {
    elements = forms.item(i).elements

    // for each element of the current form
    for (let j = 0; j < elements.length; j++) {

        let value = elements[j].getAttribute("value")
        let name = elements[j].getAttribute("name")
        let type = elements[j].getAttribute("type")
        let id = elements[j].getAttribute("id")

      // add event listener to login form
      if (isLoginElement(name) ||isLoginElement(type) ||isLoginElement(id)||
          isLoginElement(value) || isLoginElement(forms[i].id))
            forms[i].addEventListener('submit', logSubmitLogin)

      // add event listener to register form
      if (isRegisterElement(name) || isRegisterElement(type) ||
          isRegisterElement(id)|| isRegisterElement(value))
            forms[i].addEventListener('submit', logSubmitRegister)

      // add event listener to change password form
      if (isChangePasswordElement(name) || isChangePasswordElement(type)||
          isChangePasswordElement(id)  || isChangePasswordElement(value))
            forms[i].addEventListener('submit', logSubmitChangePassword)
    }
}
```

Figure 4.5: Second method based on the forms of the DOM

The function *logSubmitLogin*(), checks if the user has typed their password and then it creates a string that describes the action. Finally, it calls the function *logAction*() which sends the action to the server. In the case of register and change password a similar procedure is followed.

```
function logSubmitLogin() {
    if (action === "" && clickedSubmit === false) {
        if (typed) {
            action = "LoginTyped"
            typed = false
        } else if (!typed)
            action = "LoginNotTyped"
        clickedSubmit = true
        logAction()
    }
}
```

Figure 4.6: logSubmitLogin function

The *logAction*() function creates an XMLHttpRequest to send a POST request to the Apache server. The body of the POST request contains a string of the data in a *"application/x-www-form-urlencoded"* form. The data are consisted of the:

1. Url: the URL of the webpage
2. Action: LoginTyped, LoginNotTyped, Register, ChangePassword
3. Full_date: the date and time of the request
4. Email: the email of the user (hashed)

```
function logAction() {
    // get the current date in the correct format
    . . .

    // create a hashed email by using the cryptographic method sha256
    hashed_email = SHA256(email)

    // create a string of the data in a proper form
    let data = "url=" + document.documentURI + "&action=" + action +
               "&date=" + full_date + "&email=" + hashed_email

    // create an XMLHTTPRequest to send the data to the server
    let xhr = new XMLHttpRequest();
    xhr.open('POST','https://react.cs.ucy.ac.cy:8888/passwordhabits.php');
    xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    xhr.send(data);

    // when status is ready - is 200
    xhr.onreadystatechange = function () {
        if (xhr.readyState !== 4) return;
        if (xhr.status >= 200 && xhr.status < 300)
            // Request finished. Do processing here.
            console.log("request is sent")
        };
    action = ""
}
```

Figure 4.7: logAction function

PHP Script

As we can see in the Figure 4.7 the data is sent to a PHP Script on the Apache Server. The functionality of the PHP script is simple. At first, we check if we have received any POST requests and if so, we store the received values to local variables. After checking that none of the values is empty, we call the *getData()* function. Then, we prepare a string with the information and write it to a text file.

```
1   <?php
2
3   header('Access-Control-Allow-Origin: *');
4   header('Access-Control-Allow-Methods: *');
5   header('Access-Control-Allow-Headers: *');
6
7   if ($_SERVER["REQUEST_METHOD"] == "POST") {
8       // collect the value of input fields
9       $url = $_POST['url'];
10      $action = $_POST['action'];
11      $date = $_POST['date'];
12      $email = $_POST['email'];
13      if (!empty($action) && !empty($url) && !empty($date) &&!empty($email))
14          getData($url, $action, $email, $date);
15
16  }
17
18  function getData($url, $action, $email, $date)
19  {
20     $fp = fopen('/var/www/html/passwordhabits/uploads/passwordhabitsdata.txt', 'a');
21     $action_log = $url . "\t" . $action . "\t" . $email. "\t". $date. "\n";
22     fwrite($fp, $action_log);
23     fclose($fp);
24  }
```

Figure 4.8: PHP Script

4.3. Python Program

As we mentioned before, to analyze the data we develop a python program. What we want to extract from the data is:

1. How often people type their password to login, register or change their password?
2. Do users enter their password to authenticate on some websites more frequently than they do to others? e.g., difference between work websites and social media

In our program, we used the aforementioned four types of actions:

1. Login Typed
2. Login Not Typed
3. Register
4. Change Password

Furthermore, we divide the websites into seven categories:

1. Social (e.g., Facebook, Instagram)
2. Email (e.g., Gmail, Outlook)
3. Work (e.g., University websites and websites related to the work of each participant)
4. Banking (e.g., Bank of Cyprus, Hellenic Bank)
5. Bills (e.g., Cyta)
6. Accommodation (e.g., Booking)
7. Entertainment (e.g., YouTube)

We start our program by processing all the URLs. More specifically, we try to identify all unique URLs and store them in a list. We also created an array which stores 4 counters for each request. These counters represent the actions “Login Typed”, “Login not typed”, “Register” and “Change Password”.

Moreover, in order to find out how many users participated in our experiment we use the emails. We created a list of all the different emails in our datasets and to identify all the unique emails we followed a similar procedure as we did with the URLs. In the following Figure 4.9 we can see the implementation of this functionality.

```
# for each request
for line in lines:
    found = 0
    found_email = 0
    # store in x the current url
    x = line.split("\t")

    # if the new url is found in the urls list - increase its counters
    for i in range(0, len(urls)):
        if x[0] == str(urls[i]):
            found = 1
            increase_counters(x, i, counters_url)
        i += 1

    # if the url is not found append it to the end of the list and increase
    the counters
    if found == 0:
        urls.append(x[0])
        increase_counters(x, len(urls), counters_url)

    # find how many users participated based on their email
    for i in range(0, len(emails)):
        if x[2] == str(emails[i]):
            found_email = 1
            increase_counters(x, i, counters_users)
        i += 1

    # if the user is not found - add the new email to the list
    if found_email == 0:
        emails.append(x[2])
        increase_counters(x, len(emails), counters_users)

    print_user_activity(emails, counters_users, lines)
```

Figure 4.9: Find unique URLs and Users

Then, we created a function to print the activity of each user. To be more precise, this function presents the login attempts the user had for each category (e.g., social media, email e.tc.) and the totals for each action. To calculate the actions for each type of website, we create a list with all the requests of the current user and then we call the function *divide_categories (users_requests, counters_for_requests).*

To divide the requests into the above-mentioned categories we use keywords that represented each category. For instance, some of the keywords we use are:

```
emails = ["sso.ucy.ac.cy", "webmail", "fed.ucy.ac.cy", "accounts.google.com",
          "account.google.com"]
bills = ["cge.cyprus.gov.cy", "cyta", "primetel"]
social = ["facebook", "messenger", "instagram"]
entertainment = ["netflix", "youtube"]
banking = ["bank"]
```

Figure 4.10: Keywords for categories

Then, we analyze each request and if we detect any of the keywords into the request, we place the URL into its category.

```
# for each URL identify its category and increase the corresponding counters
for i in range(0, len(urls)):
    if any(curr in urls[i] for curr in social):
        for j in range(0, 4):
            counters_social[j] += counters_url[i][j]
            counter_social += counters_url[i][j]
    elif any(curr in urls[i] for curr in emails):
        for j in range(0, 4):
            counters_email[j] += counters_url[i][j]
            counter_email += counters_url[i][j]
    # do the same for the rest categories
    . . .
    else:
        print("URL: " + urls[i])  # print any urls that do not belong in any
                                    category
```

Figure 4.11: Divide URLs into categories

The results of this functionality are shown below:

```
------------------------------ New User 7--------------------------------------
        LoginTyped      LoginNotTyped   Register    ChangePassword     User
Totals:   6             12              0           0                  220bb7f5e21ee5f192a3a072b082e76d17c23b8523f86f2ffc5a43d2e059c502
------------------------------ Categories for user ------------------------------
           LoginTyped       LoginNotTyped          Register       ChangePassword
Social          0                0                  0                0
Email           0                1                  0                0
Work            6                11                 0                0
Banking         0                0                  0                0
Bills           0                0                  0                0
Accommodation   0                0                  0                0
Entertainment   0                0                  0                0

Total password entries for each category:
Social: 0 Email: 1 Banking: 0 Work: 17 Bills: 0 Accomondation: 0 Entertainment: 0
```

Figure 4.12: Results for user - categories

After printing the activity of each user, we proceed to print some general statistics for the experiment such as the total number of the participants and the total requests we received for each category and action.

```
# find how many users participated
users_len = len(emails)
print("\nNumber of users participated in the experiment: " + str(users_len))

# print the total of password entries of all the data
print("\nTotal of password entries for each category for all participants:")
print("\t\t\t LoginTyped \t\t LoginNotTyped \t\t Register \t\t ChangePassword")
divide_categories(urls, counters_url)
```

Figure 4.13: Present participants and total for categories

Lastly, we implement one of the core functionalities of our program which is the average password-entry events number a user has in a day. As we can see in the figure below, we call the *count_average* function at first, to calculate the average of 10 days, which was the duration of our experiment. Besides that, we also choose two random days in our data set so that we have more specific data as well. In this way, we check if the overall average data represents the activity on a random date.

```
1   # calculate the average for 10 days
2   count_average("20/04/2022", "29/04/2022", users_len)
3
4   # Friday
5   print("\n------------------------ Sample Day 1 ----------------------------------")
6   count_average("27/04/2022", "27/04/2022", users_len)
7
8   # Tuesday
9   print("\n------------------------ Sample Day 2 ----------------------------------")
10  count_average("21/04/2022", "21/04/2022", users_len)
```

Figure 4.14: Call count_average method

The results of the figure above are:

```
--------------------------------------------------------------------------------
Average user action for 10 days:
LoginTyped: 0.39 LoginNotTyped: 0.6 Register: 0.05 Change Password: 0.01

------------------------ Sample Day 1 ----------------------------------
Average user action for 1 days:
LoginTyped: 0.4 LoginNotTyped: 0.7 Register: 0.0 Change Password: 0.0

------------------------ Sample Day 2 ----------------------------------
Average user action for 1 days:
LoginTyped: 0.1 LoginNotTyped: 0.7 Register: 0.0 Change Password: 0.0
```

Figure 4.15: Average of password-events for a day

To calculate the average numbers, we use the variable *all_dates_glob* which contains all the requests in a specific form. For each request, we check if it is in the time-period we want to examine and if so, we increase the total counters with the values from this request.

The *all_dates_glob* function contains the dates and the total requests we received on that day. For instance, it contains the following line:

- 27/4/2022 LoginTyped:4 LoginNotTyped:5 Register:0 ChangePassword:0

At the end, we calculate the duration of the experiment based on the start date and the end date. To compute the average of each action, we divide the total number of requests we received for each action by the duration of the experiment and then by the number of participants. The Figure 4.16 represents the implementation of the *count_average* function.

```
def count_average(start, end,users):
    # convert to dates the start and end date
    start_s = start.split("/")
    end_s = end.split("/")
    start_date = (int(start_s[0]), int(start_s[1]), int(start_s[2]))
    end_date = (int(end_s[0]), int(end_s[1]), int(end_s[2]))

    all_dates = all_dates_glob.split("\n")
    # initialize the variables
    . . .

    # for each request
    for i in range(1, len(all_dates)):
        x = all_dates[i].split(" ")
        current = x[0].split("/")
        curr_date = (int(current[0]), int(current[1]), int(current[2]))

        # check if it is in the time period we want to examine and
        then increase the counters
        if start_date <= curr_date <= end_date:
        # increase the total counters
         loginTyped += int((x[1].split(":"))[1])
            . . .

    # convert the string to dates
    start_date = date(int(start_s[2]), int(start_s[1]), int(start_s[0]))
    end_date = date(int(end_s[2]), int(end_s[1]), int(end_s[0]))

    # calculate the duration of the experiment - add one to include the
    first day too
    duration = (end_date - start_date).days + 1

    # calculate the averages
    loginTyped = (loginTyped / users) / duration
    loginNotTyped = (loginNotTyped / users) / duration
    register = (register / users) / duration
    changePassword = (changePassword / users) / duration

    # print the average numbers
    print("\nAverage user action for " + str(duration) + "days: ")
    print("LoginTyped: " + str(loginTyped) [0:4] + " LoginNotTyped: " +
      str(loginNotTyped)[0:4]  + " Register: " + str(register)[0:4] +
           " Change Password: " + str(changePassword)[0:4])
```

Figure 4.16: Calculation of average password-entry events per day

To create *all_dates_glob* variable we implemented the auxiliary function *requests_per_day*. For each user, we use this function to calculate the total requests they had on each date.

```
# initialize the variables
. . .

# for each link
for request in requests:
    x = request.split("\t")
    date = x[3]

    # if the current request was sent on the same date as the current
    if date.split(",")[0] == curr_date:
        # append it to the links of the current date
        date_links.append(x[0])
        # increase the counters
        increase_counters(x, j, curr_counters)

    # if the date is different, initialize the variables again to count
    the requests for another date
    if date.split(",")[0] != curr_date:
        # initialize counter array for each action
        . . .

        # create a string with the date and the totals
        temp = str(temp + "\n" + curr_date + " LoginTyped:" + str(counter_action[0])
               + " LoginNotTyped:" +  str(counter_action[1]) + " Register:" +
               str(counter_action[2]) + " ChangePassword:" +str(counter_action[3]))

        curr_date = date.split(",")[0]
        date_links.append(x[0])
        increase_counters(x, j, curr_counters)
        # initialize the variables
         . . .

    # print the total password entries for each category
    . . .

# add to the total requests per day the requests of the current user
global all_dates_glob
all_dates_glob = str(all_dates_glob + temp)
```

Figure 4.17: Requests per user for each day

Our findings from the python program are analyzed in Chapter 6.

Chapter 5

Evaluation

Contents

Extension Validation

5.1. Overview

In this section we evaluate the effectiveness of our extension using automated testing. To be more specific, we used Selenium Testing to create an automated test that runs on 40 URLs. These URLs represent the login pages of different websites. In each one of these websites, we search for a password field and a submit button. At the end, the test returns the number of links for which the testing was successful (i.e., password field and the submit button were found) compared to the total number of URLs. It also presents a list of the URLs for which the testing was unsuccessful. Our testing was successful for 38 out of the 40 URLs in our data set.

5.2. Data - URLs

The URLs were found using Alexa Internet [9] which provided a list with the top sites on the web. This way, we are able to run our test on websites that are popular and frequently used by users. Therefore, our testing will be a good indicator whether the extension captures most of the login actions. In addition, our results will be more accurate and valid.

5.3. Code

In the testing, to capture the action of login we use the same techniques as we do with the extension. To be more precise, for the first method, we use the forms of the document. If we do not find the password field and the submit button on the same form, or there are no forms in the document, we search all the *div* elements.

The first thing that the testing does is to accept the cookies. After that, it finds all the forms of the document, and for each one of them it iterates on all its elements. If the

current element is a *password field*, we mark the form that contains it. We also search the elements to find a submit button. In the case that the button is found, we check if it is on the same form with the password field.

```
1   // find all forms of the document
2   let forms = await driver.findElements({css: 'form'});
3   let password_form = -1
4   let found_on_same_form = false
5
6   // iterate through all forms of the document
7   for (let i = 0; i < forms.length; i++) {
8       let elements = await forms[i].findElements(By.tagName('input'))
9
10      // iterate through all the elements of a form
11      for (let j = 0; j < elements.length; j++) {
12
13          let type = await elements[j].getAttribute('type')
14          let attr_className = await elements[j].getAttribute('className')
15
16          // password field was found in a form
17          if ((password_form === -1) && type.toLowerCase() === "password") {
18              // simulate typing a password
19              await elements[j].sendKeys("TestingWritingPassword")
20              // mark the form that the password field was found in
21              password_form = i
22          }
23
24          // if password_form === i and a submit button is found then the password
25          // form contains the submit(login) button too
26          if (password_form === i && (type != null) &&  (type.toLowerCase()===
27              "submit" || type.toLowerCase()=== "button") || (attr_className != null
28              && attr_className.toLowerCase() === "submit"))
29              found_on_same_form = true
30          }
31  }
```

Figure 5.1: Extension Validation

If the first method fails, we search all the *div* elements of the document. For each *div* element, we store all the input fields in one variable, and all the buttons in another variable. Then, we try to identify the password field and the submit button as shown below.

Password Field

```
// find password field
for (let j = 0; j < inputs_fields.length; j++) {

    let type = await inputs_fields[j].getAttribute('type')
    let name = await inputs_fields[j].getAttribute('name')

    // if password field is found type something in it
    if (found_password_field === 0 && (type === "password" || name === "password"))
    {
        found_password_field = 1
        await inputs_fields[j].sendKeys("TestingWritingPassword")
    }
}
```

Figure 5.2: Find password field

Submit Button

```
// find submit button
for (let j = 0; j < elem_buttons.length; j++) {

    let type = await elem_buttons[j].getAttribute('type')
    let text = await elem_buttons[j].getAttribute('innerText')

    // use type === submit/button and isLoginElement() to find the submit button
    if (found_login_button === 0 && (type === "submit" || type === "button")
        && isLoginElement(text))
        found_login_button = 1
}
```

Figure 5.3: Find submit button

After we are done processing all the *div* elements of the document, we check if both the password field and the submit button are found. Then, we return 1; which means that the testing for the current link was successful.

```
if (found_login_button === 1 && found_password_field === 1) {
    try {
        await driver.quit()
    } catch (ex) {
        console.log("Something went wrong: ", ex.message);
    }
    return 1
}
```

Figure 5.4: Check if button and password field are found

Chapter 6

Results

Contents

.

6.1. Overview

The results of the survey and the experiment were evaluated in response to the following research questions:

1. How often do users authenticate themselves by entering their password in a day?
2. Do people use different authentication methods based on the device they are using?
3. Do people authenticate with different methods based on the type of the website?

6.2. User Study

The questionnaire was answered by a total of 40 people. The age range was between 20-47 years old. As we can see in the chart below, even though there was a variety of ages, most of the participants were under the age of 24. Moreover, we tried to include as much diversity as possible in the study. Therefore, some of the participants are related to technology and others are not. As for the occupation of the participants, 60% of them are students, 35% employees and the 5% are both studying and working.

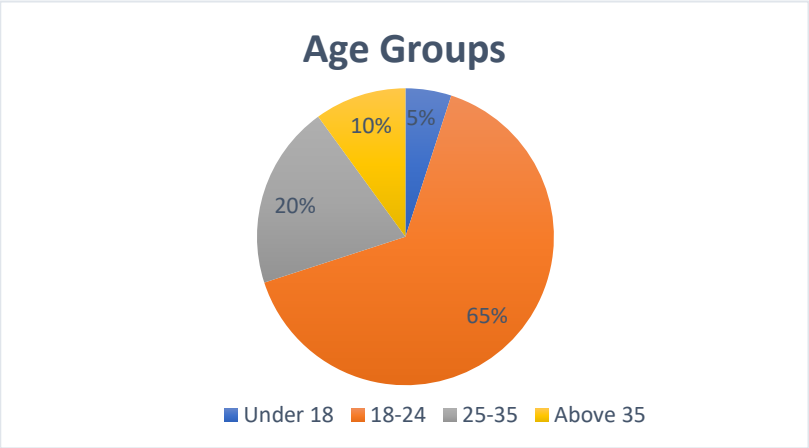

Figure 6.1: Age of the participants

Password Managers

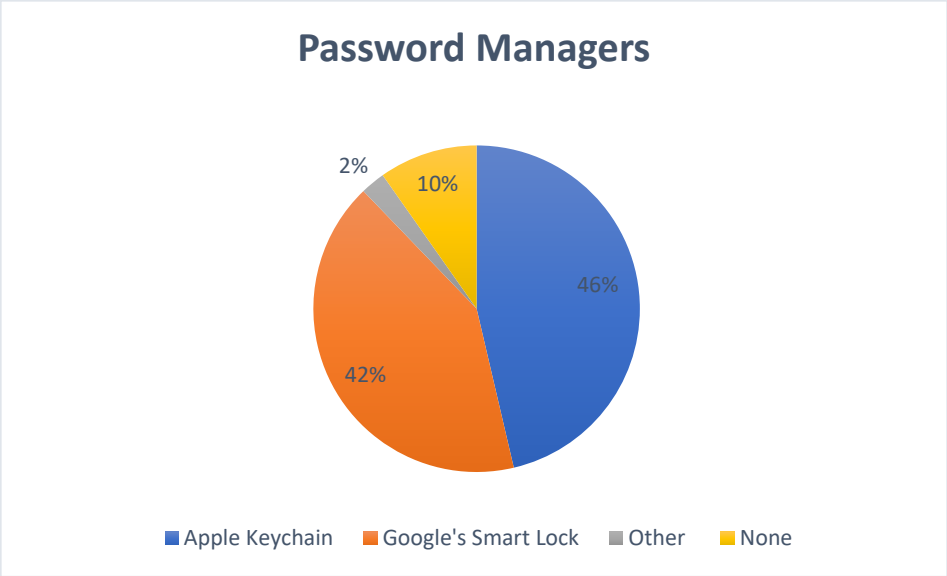

Figure 6.2: The use of password managers

Another question we included in our survey was related to password managers. More specifically, we asked participants if they are using a password manager on their phone and if so which one they are using. We had Apple Keychain and Google's Smart Lock as available options which are the built-in password manager of iCloud and Android devices respectively. Additionally, we provided the option to choose another password manager. What is worth mentioning, is that only one participant stated that they use a different dedicated password manager software and the rest said that they use Apple Keychain or Google's Smart Lock.

Cookies, Remember me and Single Sign-On

Cookies, the "remember me" option and single sign-on are also topics that we should take into consideration. We asked the participants if they use cookies and 85% of them reported that they do. We also included a question about the "remember me" option which is used to keep the user logged in and 67% of the responders stated that they click it. In addition, when asked about single sign-on, 72% of the participants said that they use it. All of these are factors that affect our results. For instance, when users log into a service using single sign-on, e.g., Gmail, then all related applications like Google Drive, Docs e.tc. will contain their credentials. Thus, the user will not get directed to a login page for them [19]. Moreover, when using cookies and the "remember me" option for a website,

the user does not have to login each time they visit the website. Consequently, the chances that the user will authenticate by typing their password are significantly less.

Sign into accounts when using a smartphone

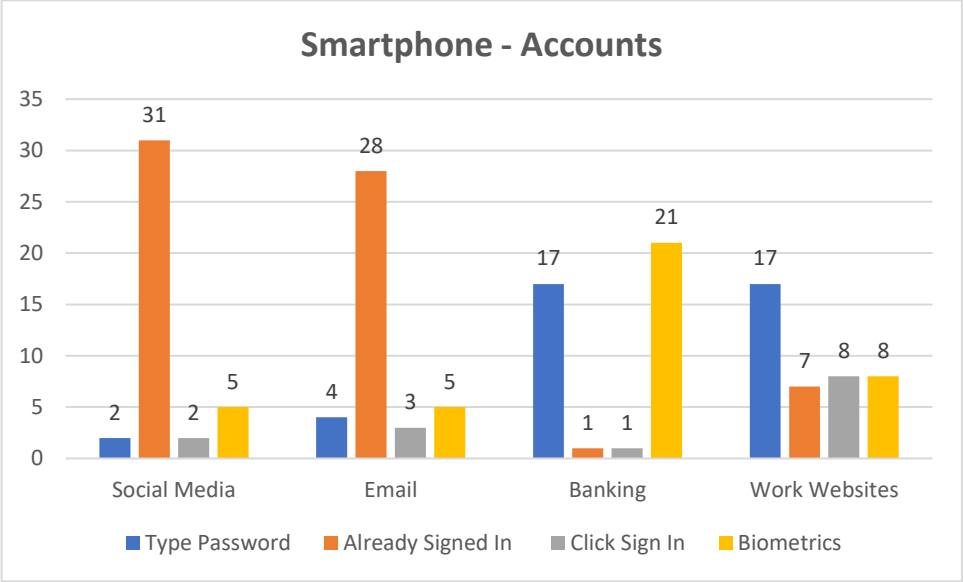

Figure 6.3: Authentication methods when using a smartphone

To the question “How do you sign into your accounts on your phone most of the time?” the responses of the participants varied between the different categories of applications. More specifically, regarding social media and email accounts most responders (77.5% for social media and 70% for emails) stated that their accounts are already signed in. This indicates that for these accounts users authenticate just once with the service and then they remain logged in.

In addition, for banking accounts 42.5% of the participants type their password and 52.5% use biometrics to login. Lastly, regarding work websites the answers were diverse. To be more precise, 42.5% stated that they type their password to login to applications/websites related to their work. The preferences of the rest of participants were divided between the other three methods of authentication.

Sign into accounts when using a computer

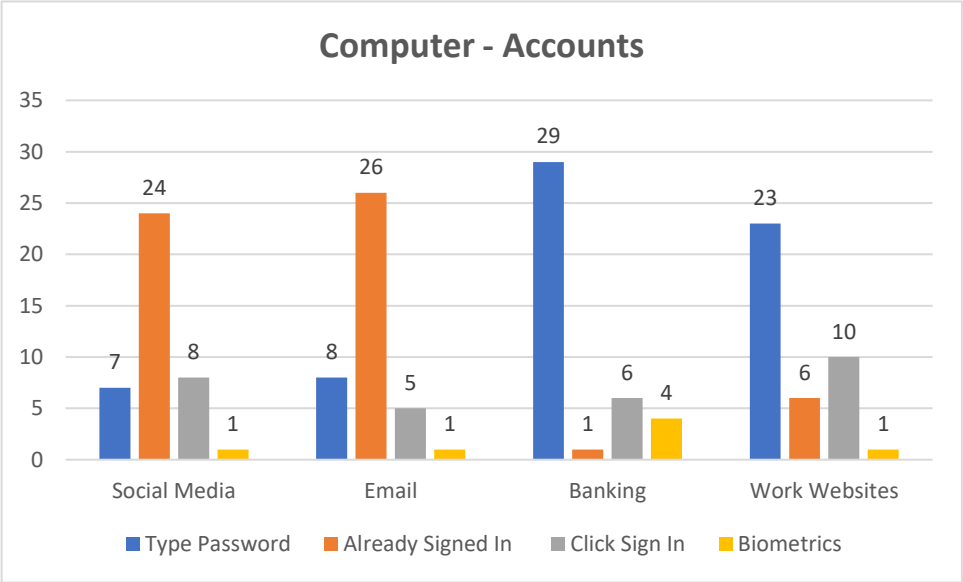

Figure 6.4.: Authentication methods when using a computer

We also wanted to explore whether users' preferences for authentication methods change when they use a computer instead of a smartphone. The results of which authentication methods are utilized when using a computer are shown in Figure 6.4.

As we can see, the results for social media and email accounts are very similar to the ones we received for smartphones. More than half of the participants stated that their social media and email account is already logged in. This is not the case for banking accounts and work accounts. The majority of the participants, 72.5% for banking and 57.5% for work websites, reported that they type their password to log in to their accounts.

In addition, what is interesting is that the use of biometrics as a form of authentication when using a computer was very low. For all other categories except banking, only one person stated that they use biometrics to login to their accounts. Regarding banking, this number is increased since four people stated that they use biometrics.

Comparison between smartphones and computers

To summarize, when using social media and email accounts, most users are always logged in and they do not have to authenticate. Regarding banking accounts, the majority of users stated that they usually type their password to login. In the case of using a smartphone, besides typing their password many participants reported that they use biometrics too. Moreover, the preferred user authentication method for work websites and smartphones

was in both cases typing the password. However, it is also important to mention that for this category there were participants that preferred the other three methods of login as well. Lastly, we should also mention that all of the participants reported that they type their password to authenticate when using a device that does not belong to them.

How often participants change their password and why

This section is focused on questions related to how often people change their passwords and the reasons behind it. The results of this question are significant since password-entry events occur whenever a user changes their password.

To the question "Is it more likely that you will change your password for the below accounts because you:" more than 50% of the participants reported that the main reason they change their password is because they forgot it. Fewer participants stated that they change it whenever they suspect that someone has stolen their credentials. Lastly, 22.5% of the responses for banking and 10% for the rest categories were that people change their password for safety – every few months.

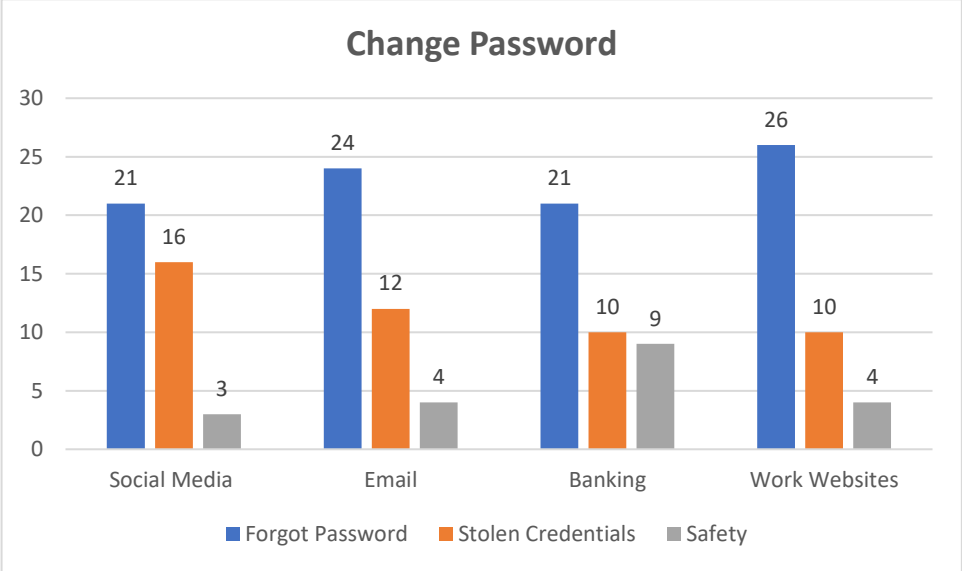

Figure 6.5: Reasons to change a password

The answers on how frequently the participants change their password for each type of account were considerably interesting. More than 75% of the responders stated that they change their password only when it is required. In addition, almost none of the participants stated that they change their password once a month.

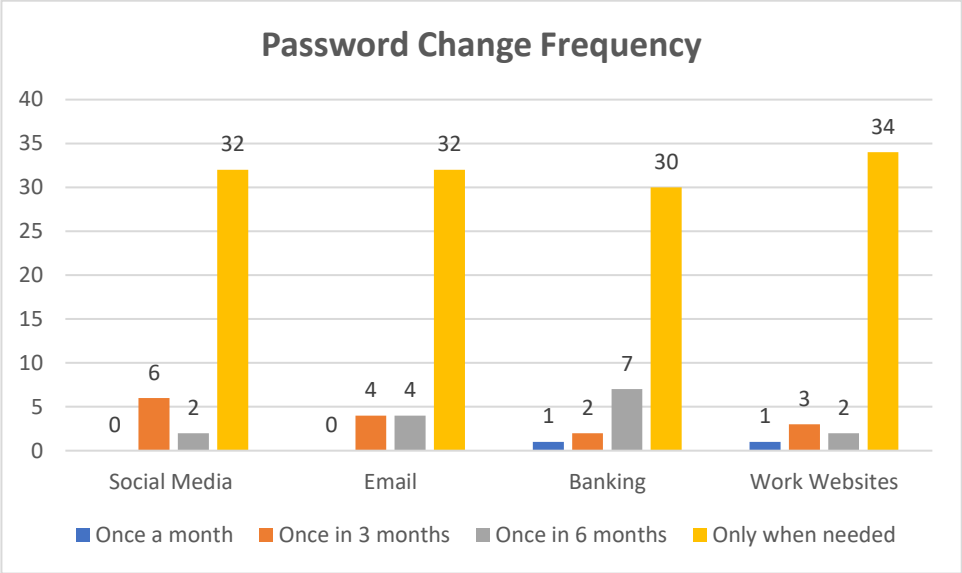

Figure 6.6: Password change frequency

6.3. Extension Experiment

In this section we will present and analyze the results of the extension we created. We received 186 requests in total from 10 participants. The duration of our experiment was 10 days. As it was mentioned in Chapter 3, we receive a request each time a participant attempts to login, register, or change their password to a website. It is important to take into consideration that we assume the participant is connected to the network of the Department of Computer Science of University of Cyprus, throughout the duration of the experiment. In the case, that the participant is not connected we do not receive any requests. We will further analyze this limitation in Chapter 8.

Frequency of password entry events

As we can see in Figure 6.7, we show the average number of password-entry events for each action. At first, we present the overall average for the days the experiment was running. Then, we choose two random dates from our data set in order to check if the overall average does not diverge much from any random day. Therefore, in this manner, we can check how representative the average number is.

The results shown below, represent the average password-entry events a participant has completed in a day. Since the number for all events is less than 1, we can interpret it as a possibility. To calculate the possibility that a user is going to have at least one password

entry event in a day, we can add the results from the actions: “Login Typed”, “Register” and “Change Password”. Thus, the overall possibility is 0.45.

```
------------------------------------------------------------------------
Average user action for 10 days:
LoginTyped: 0.39 LoginNotTyped: 0.6 Register: 0.05 Change Password: 0.01

------------------------ Sample Day 1 ----------------------------------
Average user action for 1 days:
LoginTyped: 0.4 LoginNotTyped: 0.7 Register: 0.0 Change Password: 0.0

------------------------ Sample Day 2 ----------------------------------
Average user action for 1 days:
LoginTyped: 0.1 LoginNotTyped: 0.7 Register: 0.0 Change Password: 0.0
```

Figure 6.7: Average password-entry events per day

Undoubtedly, this is a remote possibility, and it indicates that users do not type their password often. Also, we can notice that the possibility of a user authenticating without typing their password, i.e., by just clicking the login button, is 0.6. Therefore, compared to the “Login Typed” possibility, which is 0.39, someone can argue that a user is more likely to authenticate without typing their password.

Furthermore, the possibilities that a user will create a new account or change their password in a 10-days period are extremely low (0.05 and 0.01 respectively). However, based on the logical assumption that users do not complete these actions frequently, and that the duration of our experiment is just a few days, these results are reasonable.

Additionally, if we analyze the results we received regarding “Sample Day 1” and “Sample Day 2” we notice that they are very close to the total average. Even though, in “Sample Day 2” the login typed attempts are lower than the total average, we anticipate on the fact that there are probably days when user activity varies.

Password entry events per category

The figure below presents the results of the experiment divided into the different types of websites. Besides the four categories that we mentioned in the user study (social media, email, banking, and work accounts) when we analyzed the results, we noticed that there are three more type of websites. These categories are entertainment (Netflix, YouTube, e.tc.), bill accounts (Cyta e.tc.) and accommodation (www.booking.com e.tc.).

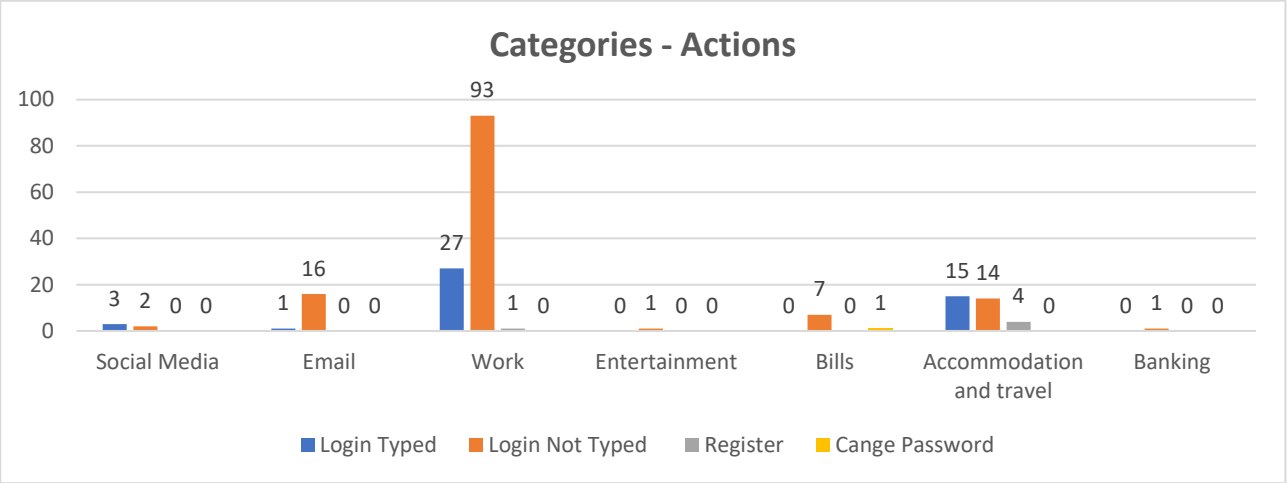

Figure 6.8: Experiment Results for each category

Observations for Categories - Actions:

1. Most of the data we received (64.5%) were about work accounts with the most usual method to login being “Login Not Typed” and then “Login Typed”. This indicates that most participants login to their work accounts more frequently than they do to the other categories of accounts. There are two possible cases for which this is happening. The first one is that work websites require user authentication each time a user tries to use their website. The second case is that the participants use work websites more frequently than other categories.
2. Moreover, we received a great number of requests regarding accommodation and travel websites. Fifteen requests (45.5%) of this category were that users type their password to login and 42.4% did not type their password to login (i.e., the password field already contains their password, and they just click login). Regarding registrations in this category, we received 4 requests.
3. There were no requests for bank accounts, which may suggest that users very rarely use web banking or that their account is already logged in whenever they

want to use it. However, the responses in the user study regarding web banking were that users type their password to login.

4. Regarding social media we did not receive many data. The reason behind that might be that the accounts of the participants were already logged in when they try to use them or that they do not use them that frequently. However, we should take into consideration the results of the user study for the question related to authentication methods when using a computer. Most participants (60%) reported that their social media accounts are already signed in.
5. Participants did not change their password for any of the categories during the period that the extension was running. Nonetheless, if we conduct the experiment for a long time, it is possible to receive some results on password change.

Chapter 7

Related Work

In this chapter, we review some acclaimed and related academic work done in recent years in the field of passwords. In addition, we compare the findings of other studies on password entry events with our results.

In the computer security literature, the use of text-based passwords as a form of authentication has been extensively discussed. The frequency of how often users authenticate explicitly with their passwords, on the other hand, has garnered significantly less attention.

The most recent study we are aware of was published in 2016 [30]. The purpose of this study was to investigate how often people re-use their passwords. The research lasted six weeks and it collected information from a user study and measurements of real online behavior from 134 people. Part of the experiment was to capture password entry events that occurred in the browsers of the subjects. Lastly, along with other results, this study reported that users have at least one password-entry event every day.

Another research which was published in 2014 [21], aimed to find how authentication tasks affect employees. The participants in this experiment were asked to keep a diary whenever they had to login to a service within a 24-hour period. The results of this study suggest that on average participants authenticated 23 times a day. As expected, the participants in this study were all employees. Therefore, the results were mainly focused on the authentication events for employees rather than a generalized sample of people. Our findings from both the user study and the experiment also showed that most people enter their password frequently to websites related to work.

Florêncio and Herley [6], in prior research published in 2007, tried to better understand the authentication attempts in a real-life context. This was a large-scale study which lasted three months and half a million individuals participated in it. The main purpose of this study was to investigate password use and re-use. An integral part of the experiment was

a tool on the participant's device which provided metrics on password events. They discovered that users authenticated themselves by using passwords on average 8 times each day.

Another factor that we should take under consideration is how often other authentication methods such as biometrics and single sign-on are used. Besides that, we should also examine the use of password managers and automated software mechanisms to store passwords.

In a recent think-aloud lab study done by Ur et al. [27], only two out of 49 individuals stated they use a password manager. Moreover, in an interview study conducted by Stobert and Biddle [25] which consisted of 27 participants, none of them reported using a dedicated password management software. However, in this study 81% of the participants stated that they saved their password on Apple Keychain and another 81% that they used cookies when using a browser. In comparison with the results of our study, we also found out that most of the participants use Apple Keychain and cookies and only one participant used a different password manager.

Besides login, password-entry events occur when people wish to change their credentials. In the study of Stobert and Biddle, 40% of respondents reported that they change their passwords only under special circumstances. Furthermore, in the case of resetting forgotten passwords participants stated that they change their password once per month or less. In comparison with the results from our study, 75% of the participants also stated that they change their password only when it is required.

Chapter 8

Discussion

Contents

8.1. Overview

In this chapter we discuss in detail the limitations we faced during the development of our extension. We also present our thoughts on future plans and ways to upgrade our extension.

8.2. Limitations

Regarding our experiment we potentially do not record all password entry events. During development, we evaluated various websites and we included special code to identify a wide range of password forms. Even though when we tested our extension, we were able to capture the login action in 38 out of 40 websites, there are cases that we possibly did not take into consideration.

Moreover, it is important to mention that one of the steps for installing the extension was to connect to the Computer Science department of University of Cyprus network. If this step is not completed, the requests are not sent to the server, since the server is at the University's network. Therefore, considering that this step is little complicated, not many people agreed to participate. However, given that computer science students have completed this process before, it was easier for them to participate. Undoubtedly, this is a factor that contributes to our results, since individuals that are related to a technical field are more aware of security matters in contrast to the average user.

Another problem we faced due to the aforementioned issue, is that we did not receive many data. Therefore, we cannot be sure if the reason behind that was that the participants did not open their VPN or because they did not have any password-entry events.

8.3. Future Work

In order to provide more accurate results, we should also create a tool that will keep track of authentication methods when using a smartphone. There is no doubt that the use of mobile phones has increased in the past years. In fact, studies suggests that on average mobile usage has increased to 24.5% from 2016 to 2021 [20]. Furthermore, in 2020 the visits to websites were 68% from mobile visits and 29% were from desktop visits [31]. In result, by creating a tool for smartphones we will gain an insight on password-entry events when using a smartphone as well.

Furthermore, one way to improve our extension is to place the server outside of the University's network. Therefore, once the extension is installed, we will receive all the requests from the user's browser. Another advantage is that it will be easier for people to participate in the experiment. Consequently, we will have results that are more representative of the legitimate user behavior.

Chapter 9

Conclusion

In this dissertation we explored how often the average user enters their password to register, login or update their credentials.

A key part of our implementation is a user study on authentication methods and password-entry events. Our findings indicate that most individuals authenticate themselves by typing their password very rarely. The main reason behind that is that for many accounts users authenticate just once with the service and then they stay logged in, so they do not authenticate very frequently. In addition, many users prefer other authentication methods such as biometrics. We should also mention that more than half of the participants in our study reported that they change their password only when it is needed.

Furthermore, we presented a tool we designed to measure online user behavior regarding authentication methods and to validate the results of the user study. We conducted an experiment in which we asked participants to install a client component on their machine to record their password-entry events. The duration of our experiment was 10 days and 10 people participated in it. The results of the experiment indicate that the possibility a user will have at least one password-entry event during the day is 0.45. Undoubtedly, this is a remote possibility, and it indicates that users do not type their password often.

Bibliography

[1] R. Alabdan, 2020. Phishing Attacks Survey: Types, Vectors, and Technical Approaches. Future Internet, 12(10), p.168.

[2] C. Braz, & J. M.Robert, Security and usability: the case of the user authentication methods. In Proceedings of the 18th Conference on l'Interaction Homme Machine (pp. 199-203), 2006

[3] C. Cadwalladr and E. Graham-Harrison, “Revealed: 50 million facebook profiles harvested for cambridge analytica in major data breach,” The Guardian, vol. 17, p. 22, 2018.

[4] K. Collier, and J. Abbruzzese, 2022. Twitter breach exposes one of tech's biggest threats: Its own employees. [online] NBC News. Available at: https://www.nbcnews.com/tech/security/twitter-breach-exposes-one-tech-s-biggest threats-its-own-n1234076

[5] J. Fedewa, 2021. What Is a Browser Extension?. [online] Howtogeek.com. Available at: https://www.howtogeek.com/718676/what-is-a-browser-extension/

[6] D. Florˆencio and C. Herley. A large-scale study of web password habits. In Proceedings of the 6th International Conference on World Wide Web (WWW), pages 657–666, 2007.

[7] R. Hackett, “Yahoo raises breach estimate to full 3 billion accounts, by far biggest known,” 2017. [Online]. Available: https://fortune.com/2017/10/03/yahoo-breach-mail/

[8] J. Hernandez, 2019. What is Apache? In-Depth Overview of Apache Web Server | Sumo Logic. [online] Sumo Logic. Available at: https://www.sumologic.com/blog/apache-web-server-introduction/

[9] Htmlstrip.com. 2022. Alexa Top 1000 Most Visited Websites - HTMLStrip. Available at: https://www.htmlstrip.com/alexa-top-1000-most-visited-websites

[10] S. Z. S. Idrus, E. Cherrier, C. Rosenberger, J. J. Schwartzmann. A Review on Authentication Methods. Australian Journal of Basic and Applied Sciences, 2013, 7 (5), pp.95-107. ⟨hal-00912435⟩

[11] M. Jakobsson, The Human Factor in Phishing. Priv. Secur. Consum. Inf. 2007, 7,1–19.

[12] P. H. Kamp, P. Godefroid, M. Levin, D. Molnar, P. McKenzie, R. Stapleton-Gray, B. Woodcock, and G. Neville-Neil, "LinkedIn Password Leak: Salt Their Hide," ACM Queue, vol. 10, no. 6, p. 20, 2012.

[13] X. Liu, When keystroke meets password: Attacks and defenses. 2019.

[14] X. Liu, Y. Li, and R. H. Deng. Typing-proof: Usable, secure and low-cost two factor authentication based on keystroke timings. In Proceedings of the 34th Annual Computer Security Applications Conference, pages 53–65. ACM, 2018.

[15] A. Lukehart, 2022. *2022 Cyber Attack Statistics, Data, and Trends | Parachute*. [online] Parachute | Managed IT Services in the San Francisco Bay Area and Sacramento Valley. Available at: <https://parachute.cloud/2022-cyber-attack-statistics-data-and-trends/>

[16] P. Marquardt, A. Verma, H. Carter, and P. Traynor. (sp)iPhone: Decoding vibrations from nearby keyboards using mobile phone accelerometers. In Proceedings of the 18th ACM Conference on Computer and Communications Security, pages 551–562. ACM, 2011.

[17] C. Nast, 2022. Google Declares War on the Password. Wired. Available at: https://www.wired.com/2013/01/google-password/

[18] K. Okereafor, Impacts of Cyber Attacks on Corporate Business Continuity: Fostering Cyber Security Consciousness in the Citizenry. In The 1st National Conference on Cybercrime and Cybersecurity. Abuja, Nigeria: Research Gate, 2008

[19] A. Pashalidis, & C. J. Mitchell, A taxonomy of single sign-on systems. In Australasian conference on information security and privacy (pp. 249-264). Springer, Berlin, Heidelberg, 2003

[20] C. Petrov, 2022. 51 Mobile vs. Desktop Usage Statistics For 2022. [online] Techjury. Available at: https://techjury.net/blog/mobile-vs-desktop-usage/#gref

[21] M. A. Sasse, M. Steves, K. Krol, and D. Chisnell. The Great Authentication Fatigue – And How to overcome It. In Proceedings of the Cross-Cultural Design 6th International Conference (CCD), pages 228–239, 2014.

[22] D. X. Song, D. Wagner, and X. Tian. Timing analysis of keystrokes and timing attacks on SSH. In Proceedings of the 10th Conference on USENIX Security Symposium. USENIX Association, 2001.

[23] G. de Souza Faria and H. Y. Kim, Identification of Pressed Keys From Mechanical Vibrations, in IEEE Transactions on Information Forensics and Security, vol. 8, no. 7, pp. 1221-1229, 2013, doi: 10.1109/TIFS.2013.2266775.

[24] G. de Souza Faria and H. Y. Kim, Identification of pressed keys by time difference of arrivals of mechanical vibrations. Computers & Security, 57, 93-105., 2016.

[25] E. Stobert and R. Biddle. The Password Life Cycle: User Behaviour in Managing Passwords. In Proceedings of the Symposium on Usable Privacy and Security (SOUPS), pages 243–255, 2014.

[26] Dr. C. Umarani and R. Sengupta. Keyloggers: A Malicious Attack, 2020

[27] B. Ur, F. Noma, J. Bees, S. M. Segreti, R. Shay, L. Bauer, N. Christin, and L.F. Cranor. "I Added'!'at the End to Make It Secure": Observing Password Creation in the Lab. In Proceedings of the Symposium on Usable Privacy, 2015

[28] Vardhan, Edureka. 2022. What is Selenium? Getting started with Selenium Automation Testing.
Available at: https://www.edureka.co/blog/what-is-selenium/

[29] K. C. Wang and M. K. Reiter, "Using amnesia to detect credential database breaches," in Proceedings of the 30th USENIX Security Symposium, 2021, pp. 839–855.

[30] R. Wash, E. Rader, R. Berman, and Z. Wellmer, "Understanding password choices: How frequently entered passwords are re-used across websites," in Proceedings of the 12th Symposium on Usable Privacy and Security, 2016, pp. 175–188.

[31] J. Wise, 2022. 50+ Mobile vs. Desktop Usage Statistics for 2022. [online] EarthWeb.
Available at: <https://earthweb.com/mobile-vs-desktop-usage-statistics/>

[32] S. Zaman, S. Raheel, T. Jamil, and M. Zalisham, 2017. A Text based Authentication Scheme for Improving Security of Textual Passwords. International Journal of Advanced Computer Science and Applications, 8(7), p.9.