Password cracker: a test of the reliability of passwords for online security

Tuesday 25th March, 2025 - 09:12

Ivaylo Krumov
University of Luxembourg
Email: ivaylo.krumov.001@student.uni.lu

This report has been produced under the supervision of:
Benoit Ries
University of Luxembourg
Email: benoit.ries@uni.lu

Abstract

The purpose of this Bachelor Semester Project is to analyze the current state of passwords that people use on the Internet and to assess to what extent they can be deemed reliable in modern days. To achieve this, a number of related academic papers, all of which yield results from their own respective studies, have been utilized as support. Moreover, a program simulating a tool attempting to guess user-entered passwords has been produced to serve as a visual example to this research. All accomplished results of the project are thoroughly presented in this report. The ultimate goal of the report and the project as a whole is to incentivize users online to be more aware of their own practices when creating and managing passwords.

1. Plagiarism statement

I declare that I am aware of the following facts:

- As a student at the University of Luxembourg I must respect the rules of intellectual honesty, in particular not to resort to plagiarism, fraud or any other method that is illegal or contrary to scientific integrity.
- My report will be checked for plagiarism and if the plagiarism check is positive, an internal procedure will be started by my tutor. I am advised to request a pre-check by my tutor to avoid any issue.
- As declared in the assessment procedure of the University of Luxembourg, plagiarism is committed whenever the source of information used in an assignment, research report, paper or otherwise published/circulated piece of work is not properly acknowledged. In other words, plagiarism is the passing off as one's own the words, ideas or work of another person, without attribution to the author. The omission of such proper acknowledgement amounts to claiming authorship for the work of another person. Plagiarism is committed regardless of the language of the original work used. Plagiarism can be deliberate or accidental. Instances of plagiarism include, but are not limited to:
  1) Not putting quotation marks around a quote from another person's work
  2) Pretending to paraphrase while in fact quoting
  3) Citing incorrectly or incompletely
  4) Failing to cite the source of a quoted or paraphrased work
  5) Copying/reproducing sections of another person's work without acknowledging the source
  6) Paraphrasing another person's work without acknowledging the source
  7) Having another person write/author a work for oneself and submitting/publishing it (with permission, with or without compensation) in one's own name ('ghost-writing')
  8) Using another person's unpublished work without attribution and permission ('stealing')
  9) Presenting a piece of work as one's own that contains a high proportion of quoted/copied or paraphrased text (images, graphs, etc.), even if adequately referenced

  Auto- or self-plagiarism, that is the reproduction of (portions of a) text previously written by the author without citing that text, i.e. passing previously authored text as new, may be regarded as fraud if deemed sufficiently severe.

2. Introduction

Ever since the rise of the Internet, the potential threat of digital data theft has been a reason for the necessity of preventing this kind of information from falling into the wrong hands. This is exactly what has shaped modern cybersecurity over the past couple of decades into the enormous field of Computer Science it has become today.

Today, more than ever, people need to protect well their personal data on the Internet. The information that websites store for their own use is prone to all kinds of malicious attacks from hackers. Nowadays websites implement different countermeasures in order to prevent such threats. In fact,

online data protection is so significant that it has become unthinkable for a data-storing website not to utilize some sort of security mechanism. And yet, while there exist plenty of sophisticated ways to secure data, arguably none is more popular than the simple usage of passwords.

It is undeniable that the concept of password protection has gained immense popularity as a means of data security all around the world. There is a good reason for that - while being extremely basic and easy for a person to understand, it can also lead to remarkably effective results. But the simplicity and popularity of passwords can also be their biggest weaknesses. For years hackers have been using tools and methods with malicious intent to target passwords and compromise personal user accounts on the Internet. These hacker attacks on passwords have become frequent in modern days and thus have brought the subject of the reliability of passwords up for discussion among our society.

This report presents an in-depth response to the question of password reliability. The subject of password reliability is thoroughly analyzed and the research of several studies is used to support the claim that passwords can often be unreliable. Moreover, a simple program that simulates an algorithmic password-cracking process is described. This program aims to show the susceptibility of passwords to hacker attacks and to implicitly encourage people to think about creating better passwords for protecting their personal data.

3. Project description

3.1. Domains

3.1.1. Scientific.

- **Cybersecurity.** Cybersecurity is a field of Computer Science that focuses on the utilization of technologies and digital security methods in order to protect computer systems, computer networks and digital data from unauthorized access. One of the most notable subdomains of cybersecurity is password protection. More often than not, passwords are associated with the key role they play in protecting accounts on the Internet from unwanted access.
- **Computer hacking.** Computer hacking, or simply hacking, is the act of breaching digital security mechanisms in order to access computer systems, computer networks and digital data. Usually, hacking refers to the malicious intent of compromising the aforementioned targets and taking advantage of this in an unethical or even illegal way. It is very common for hackers to target passwords of other users' accounts on the Internet so that they can access their stored personal data for their own use. There are plenty of methods in which this can be achieved, a notable example being the use of specific tools and password-finding algorithms.

3.1.2. Technical.

- **Python.** Python is the programming language of choice for the realization of the technical part of the project. The language is relatively easy to learn as a new developer and at the same time it is powerful enough to allow experienced programmers to create large and complex projects. Because of this, Python is one of the most used programming languages in the world. As a result, there exist numerous plugins and libraries that shape it as a vastly versatile language.
- **PySimpleGUI.** PySimpleGUI is a custom Python library that allows for the development of graphical user interfaces (GUIs). It uses some popular GUI frameworks like tkinter and Qt for Python as a basis to form a simpler architecture for GUI creation, making it easy to understand for beginners. Nevertheless, PySimpleGUI is powerful enough to be used for the development of simple and complex GUIs alike while still retaining accessibility to developers of all levels.
- **Visual Studio Code.** Visual Studio Code is a lightweight and versatile code editor. It supports the use of multiple programming languages (including Python), helpful tools and a plethora of extensions, which makes it customizable to every developer's unique needs. Its valuable features and simplistic visual design make it easy to use but also flexible enough to be utilized in the realization of large-scale projects.

3.2. Targeted Deliverables

3.2.1. Scientific deliverables. The question "How reliable are passwords for online security?" is answered in three steps:

- First, the question is logically split into three subquestions;
- Then, each of the subquestions is thoroughly answered;
- Finally, the answers to the subquestions are used to form a definitive answer to the main question.

3.2.2. Technical deliverables. A program simulating a password-cracking process is presented in the following way:

- Its implemented features are described
- The more interesting parts of the source code are explained
- The main encountered problems that affected its development are briefly mentioned

4. Pre-requisites

The complete realization of this Bachelor Semester Project, both the scientific and the technical part, takes advantage of several skills and competencies that the author already is in possession of before the initialization of project research. These pre-requisites are respectively identified in the 2 subsections that follow.

4.1. Scientific pre-requisites

The author understands the purpose and importance of passwords in the context of online secuirty. He is also able to comprehend information from scientific sources and capable of writing well-structured pieces of work.

4.2. Technical pre-requisites

The author has previous experience with programming in multiple languages. More particularly for this project, he is familiar with the basic concepts of programming using the Python language. As a result of the author's overall programming knowledge, he is aware of the coding obstacles that could occur during production and is prepared to improve and optimize long and complex code.

5. How reliable are passwords for online security?

5.1. Requirements

The scientific deliverable of this Bachelor Semester Project aims to satisfy several requirements, which are presented in this section. In order to provide a better quality for the structure of their presentation, they are split into two different categories, namely functional and non-functional requirements. The functional requirements are described first, followed by a description of the non-functional ones.

Functional requirements:
The scientific deliverable shall give a complete answer to the scientific question of the project - "How reliable are passwords for online security?". To do so, the question shall be divided into three subquestions to be answered using the information provided in the main scientific papers that have been chosen for the research purposes of the scientific deliverable. Some information contained in the other papers shall also be utilized as support, wherever that is suitable. The three aforementioned subquestions are the following:

- What is a password?
- What is online security?
- What is reliability?

The answers to these subquestions, when compiled together, shall logically form a definitive answer to the main scientific question.

Non-functional requirements:
The scientific deliverable should be presented in such a way that it is understandable for an audience on the level of a BiCS Semester 1 student. Essentially, this means that the use of complex terminology related to the topic of the scientific question should be avoided. In cases where this is not possible, the used word or phrase should be explained as concisely as possible in a brief and simple manner. In regard to the overall structure of the scientific deliverable, its contents should logically flow well together. Furthermore, wherever citations from other academic papers are made, credit should always be given to the referenced material and its author.

5.2. Design

The scientific deliverable is a text that aims to answer the scientific question by synthesizing useful information present in several distinct scientific sources. These sources are academic papers which have been found via research on Google Scholar. This research is independently conducted over the course of four weeks. During this time, numerous academic papers have been examined in terms of content related to the scientific question. While dozens of such articles have been found in Google Scholar searches, only a handful are recognized as containing suitable information for this Bachelor Semester Project.

Several criteria are taken into account for determining appropriate pieces of work and their selection. These are the following (in no particular order):

- Relevance of the studied topic to the topic(s) of the scientific question
- Year of publication
- Level of complexity of the used terminology and the conducted study methodology
- Overall structure and length of the paper
- Level of graphical representation of study data

The actual search process for academic papers, which precedes the selection by criteria, strictly limits itself to the use of Google Scholar and the relevant results suggested by the search engine. To look for articles, a method of searching for keywords contained in the scientific question is utilized, the most notable ones being "passwords", "security" and "reliability".

The examination of the selected articles allows to provide a full answer to the scientific question by establishing an appropriate groundwork for it, namely the answers to the three subquestions defined in the previous section of this report. For this job the contents of the scientific deliverable are logically divided in four main parts, where the first three dealing with answering a defined subquestion. Each of these parts is also logically and visually split into several paragraphs that give a clearer picture of the details needed to answer a subquestion. Finally, the text concludes with the fourth part, which is where the definitive answer to the main scientific question lies. In this last section the information from the previous three parts is used and the most relevant research results from the academic papers are analyzed in order to answer the question in a complete and thorough manner.

5.3. Production

Since the scientific question "How reliable are passwords for online security?" is expressed in such a way that it sets

its main focus on passwords, it makes the most sense to first define what a password actually is. In other words, the starting point for answering the scientific question is to find an answer to the subquestion ”What is a password?”.

Researchers are unanimous in that a password is a largely popular method of authentication [1, 2, 3, 4]. Kuka and Bahiti provide a precise scientific definition of authentication: ”any protocol or process that permits one entity to establish the identity of another entity” [5]. On the other hand, Taneski, Heričko and Brumen define authentication in a more simple manner as ”the process of verifying someone’s identity” [1]. They go on to identify three main categories into which different methods of authentication are distributed: ”knowledge-based authentication (e.g. textual or graphical passwords), biometrics authentication (e.g. retina, iris, voice and fingerprint scans) and token-based authentication (e.g. smart cards, mobile phones or other tokens)” [1]. For the purposes of this Bachelor Semester Project, the focus will be set on the first category of authentication methods and more specifically on textual passwords, which is exactly what all of the authors cited in this work have explored.

This information now makes it possible to precisely answer the question ”What is a password?”: a password is a widespread method of person identification. The word itself most commonly refers to the character string that is used in an authentication process. This means that a password can be any combination of concatenated characters (letters, numbers and/or special characters). Theoretically, there is no limit to how many or what type of characters compose a password, therefore there are infinite possibilities for passwords. For ages passwords have been used as an authentication method by people, which is indicative of their wide popularity. More recently, passwords have become a staple method of authentication in the area of computers.

Next, it would be most appropriate to provide an answer to the subquestion ”What is online security?” due to its direct relation to passwords, which were just covered in the previous paragraphs.

In a general sense, something is said to be secure when it is well-protected from undesired outside influence. In the context of computers and online environments security is usually associated with the normal flow of information processing, i.e. the undisturbed transfer and storage of different kinds of data. As already pointed out earlier, passwords are a very widespread method of authentication in all kinds of digital and non-digital environments. This is why when it comes to the topic of online security it is nearly impossible not to encounter the term ”password security”, which makes up a substantial part of one’s security on the Internet. Because of this and because passwords are the focus of this work, it makes the most sense to cover password security in more detail in order to get a better understanding of online security in general.

Chanda outlines 3 important aspects of password security, namely password storage, password generation and password theft [2, p. 24-25]. She presents 4 different ways passwords are stored on servers, each way being more technically complex than the previous. More specifically those methods are plain text storage, encryption, hashing and salted hashing. She then defines 2 separate kinds of generated passwords - human-generated and randomly generated and compares the general complexity and memorability of the two. Finally, she briefly mentions a few examples of password theft variations - hacking, social engineering, phishing, keylogging, etc. All this information provides a good overall explanation of what defines password security.

Though it is likely that Chanda’s aforementioned study on password security aspects is sufficient enough to grasp the idea of password security, it is worth adding that Ur, Bees, Segreti, Bauer, Christin and Cranor extend her definition by introducing the aspect of the security’s impact on attacks [4, p. 3749]. Essentially, they explain that a password’s resistance to malicious attacks is largely dependent on the complexity of the rules under which a specific security mechanism operates as well as the users’ own perception of a secure password. Taneski, Heričko and Brumen’s conducted research fully supports the latter, implying that users have a more important part to play in password security than they may realize [1, p. 160]. Chanda also briefly discusses this in her work [2, p. 24] but does not define it as a separate aspect of password security.

Now it is time to give an answer to the question ”What is online security?”: online security is a means to safely transfer and store different kinds of data on the Internet. The massive popularity of passwords as an authentication method has lead to password security making up a considerable part of the overall online security and therefore contributes the most to the definition and aspects of online security. Password security itself is defined by its various aspects, which include password storage, password generation, password theft and, arguably, impact on malicious attacks. Because of password security’s large influence over online security as a whole, all these concepts apply to the definition of the latter as well.

Next, there is one last remaining subquestion, that being ”What is reliability?”, to be answered. It is important that the definition of reliability is established in the context of cybersecurity, which is the domain of the main scientific question’s subject.

It is widely accepted that if someone or something is consecutively fulfilling tasks in a consistent manner, then they can be trusted to do so for future tasks of the same or similar scale and complexity. In other words, one can rely on such a person or object to provide satisfying results for each problem that is appropriately given to them to solve. So in the general case, reliability turns out to refer to the capability of being consistent with the proper completion of tasks.

In the field of computers, and more specifically in the context of passwords this definition does not change in any drastic way. Speaking about the reliability of a password, the object of interest is the password itself while the task it is expected to be consistent at fulfilling is securing data by preventing unauthorized access to it. What this really means is that a password’s reliability shows how prone it is to theft

or being bypassed, or simply said it is a test of its strength. As Chanda puts it, ”the longer it takes to break a password, the stronger it is” [2, p. 25], implying that the length of a password is an aspect of its strength, and therefore of its reliability. Moreover, Taneski, Heričko and Brumen point out that password policies, e.g. the requirement of specific types of characters, as well as password reuse largely determine a password's strength and susceptibility to attacks [1, p. 156-157].

With this information it is now possible to provide a clear answer to the question ”What is reliability?”: in the context of this project's subject, more precisely password reliability, it is defined as a measurement of a password's strength. As such, there is a very close relation between the terms reliability and strength when they are used to describe a password. Due to this, a password's reliability is heavily dependent on the elements that make up its overall resistance against threats. Namely, those dependencies are the requirements of specific policies for password creation and users' own practice of reusing the same passwords in different scenarios.

This entire in-depth analysis of the question now allows to get a clear idea of what exactly to look for when attempting to answer it. Specifically, the focus should be set on determining whether users' passwords can be trusted as a working defensive mechanism against the countless threats scattered around the immense online environment that is the Internet. To accomplish this, it would be logical to examine concrete data on online users' practices for passwords and their usage. For this purpose, the results from the study that Kuka and Bahiti have conducted [5, p. 44-47] serve as an ideal point of reference. Their data shows some of the most common poor password practices. According to their results, it is not very rare for a user to have a single password for multiple accounts or share a password with at least one other person. Furthermore, a small portion of participants admit to never changing their passwords. While this data may be of concern for the integrity of password security, it is important to mention that if compared to the results of Riley's research [6] from a decade earlier, Kuka and Bahiti's findings show a considerable improvement to the way people use and store their passwords. More precisely, this can be seen in the significant decrease of users never changing their passwords and of those who keep the same password for several accounts. And while it may seem that both studies use completely different and unrelated samples, it should be noted that for the most part both samples consist of people with enough awareness of what bad password practices mean for their security. Therefore, this comparison suggests that even though a lot of users still apply bad password-related practices to this day, there has been meaningful development in the way they think about their security online in the past 12 years.

In spite of this progress, the reality is that the data really shows that oftentimes passwords cannot truly be deemed reliable due to their unwise usage, making them and the accounts they protect more prone to malicious attacks. This vulnerability is further reflected in the meaning of passwords, for which both Kuka and Bahiti's [5, p. 44-47] and Riley's [6] studies have presented information. It turns out that more often than not, users' passwords consist of meaningful words or numbers, which can go as far as using personal information. From a psychological point of view, this is completely understandable, as these kinds of passwords are way easier for a person to remember. But in a computer security sense, as Kuka and Bahiti put it, the percentage of users who do this is ”very high, making them vulnerable as they can suffer a dictionary attack.” Nevertheless, their research also shows that a considerable number of people use passwords that are comprised of combinations of 2 or more types of characters. Most notably, passwords appear to frequently include both numbers and lowercase characters, followed by the common usage of uppercase characters. These results, however, do not fully align with the data from Riley's study [6], where participants admit to almost never using uppercase characters, suggesting that there may in fact be some improvement in password creation over the years. As for password length, Kuka and Bahiti's data [5, p. 47] reveals that there is likely no cause for concern for password security's integrity in that regard, since users generally tend to create long passwords for their accounts.

The analyzed data is sufficient to finally formulate a proper answer to the question ”How reliable are passwords for online security?”. In essence, the reliability of passwords is very much dependent on the users' practices applied to them. Generally, the more educated a person is about the importance of strong passwords for their own security, the less likely it is for them to create and manage their passwords poorly. But even this kind of awareness cannot guarantee a password is reliable enough. It is not uncommon for a password to be vulnerable to hacker attacks even among experts in cybersecurity. While conceptually as a means of data protection passwords are fine as they are, users' tendency to create passwords ignorantly and use them recklessly is what ultimately makes them vulnerable to malicious activity online, therefore unreliable for online security. And even though solutions for the limitation of bad password-related practices exist, such as utilizing password managers to avoid having to remember an array of different passwords, it all comes down to users' own responsibility to make use of them. In the end, Taneski, Heričko and Brumen's statement that ”users and their textual passwords will continue to be ”the weakest link” in any password system” [1, p. 160] appears to be an undeniable fact and it is likely to remain so for a long time.

5.4. Assessment

The aim of the scientific deliverable was to familiarize the readers with the impact passwords have on their security online by giving a complete answer to the question ”How reliable are passwords for online security?”. This has been achieved by a method of in-depth analysis of previously conducted research in the fields concerning the question, in conjunction with a synthesis of the extracted relevant information, culminating in

the formulation of the final answer.

The defined requirements, both the functional and non-functional ones, have all been met. In particular, the scientific work provides exhaustive answers to the three established subquestions necessary for a better understanding of the main question. With their help, the main question itself is also answered fully and in a thorough manner. Furthermore, the text is written in a comprehensible for a BiCS Semester 1 student way, using as little complex terminology as possible. There is also a good logical flow between all contents. Finally, all used citations from outside sources are properly credited to the respective author(s) and their academic work.

Overall, the scientific deliverable satisfies the defined requirements and the expected outcome has been achieved. The goal of presenting the necessary information to the readers has been reached and has been done so in great detail. Thus, it is expected that the aim to implicitly incentivize readers to become more aware of their password-related practices has also been fulfilled.

6. Password-cracking simulation program

6.1. Requirements

The technical deliverable of this Bachelor Semester Project aims to satisfy several requirements, which are presented in this section. In order to provide a better quality for the structure of their presentation, they are split into two different categories, namely functional and non-functional requirements. The functional requirements are described first, followed by a description of the non-functional ones.

Functional requirements:

The developed program should simulate a simplified version of a password-cracking process. To do so, it should be able to accept a password input from the user and provide output informing the user about the status of the password - either cracked or not. Furthermore, the program should simultaneously measure the time taken to crack the given password while the process is running and automatically terminate it if this time exceeds 5 minutes. Additionally, the program should give the user the ability to prematurely terminate the password-cracking process themselves, as well as the option to restart the process in case they wish to do so. Finally, each time the process is finished and the password is guessed on time, as part of the simulation the program should automatically open a text file that plays the role of an individual's personal information.

Non-functional requirements:

The program should first and foremost be easily usable by any user. The interaction between the program and the user should be simple and straightforward in order to fulfill this requirement. This should be done by creating an intuitive graphical user interface (GUI) with minimalistic design. The GUI should provide the user with a clear way to input a password, as well as present to them the program's resulting output. It should also integrate all of the other aforementioned user-controllable actions. The GUI should only visually show the most essential information about the status of the password-cracking process and not reveal unnecessary details. The process itself should be well optimized in terms of efficiency so that it takes the shortest time possible to generate the correct password.

6.2. Design

A simple program has been created as a technical deliverable for the project. It serves the purpose of simulating a password-cracking process in a way that is controlled by the user. The program produces results answering the question of whether or not a chosen password is easily crackable.

The program is designed to show its front-end to the user as a graphical user interface (GUI). The 2 windows composing the GUI are medium in size and share a common simplistic visual design with light colors, basic-looking layout and buttons that briefly change color on click. These buttons are conveniently labeled to inform the user of their purpose. Each window also has unique elements based on the functionality of that specific window, e.g. straightforward instruction text, input field, informative status text, etc.

The program has been in continuous development in the course of 5 total weeks. During this period, a remarkable amount of time has been spent on researching GUI creation in the programming language of choice, i.e. Python, in terms of finding a suitable library that could fulfill the simple visual and functional front-end needs of the program. This has been done so to make the necessary information displayed to the user by the program easy to understand. In the end, the package that has been ultimately selected allows for this goal to be achieved feasibly with a provided detailed documentation and accessible tutorials to be used as assistance.

6.3. Production

The technical deliverable is a program that consists of two main parts - the password-cracking algorithm and the graphical user interface (GUI) - which are combined to operate together. It is more appropriate to first describe the working principle of the algorithm on its own since it provides the core functionality of the program, namely the password-cracking process.

The code for the algorithm is solely written in Python. Its premise is to simulate a password attack that attempts to recover a user's password in an efficient way and use it to unlock access to some sort of data containing personal information. This is why it replicates the process of a hybrid attack, utilizing both a dictionary method and a brute-force method in succession in order to efficiently find the correct password.

Before the initialization of the password-cracking process, the algorithm is required to receive a password and store it in the memory so that it knows what its final goal is. For this

purpose, upon running the code the user is prompted to enter a password, which is then stored as a variable. Additionally, each character of the password is analyzed and depending on its type - lowercase, uppercase, digit or special character - the list of all possible characters of that type is added to a special list if it is not already in it. This special list of characters is created with the intention to be used by the brute-force part of the algorithm in case the dictionary method fails to find the password. It is important to note that the program is designed to only work with specific pre-defined characters, namely most characters in a standard QWERTY keyboard layout. This has been done in order to prevent the program from being overly complex and maintain focus on the most commonly used characters for passwords. Essentially, these include all lower- and uppercase letters from the English alphabet, all 10 digits and most of the special characters featured in the layout (Fig. 1). Whenever a password containing at least one character which has not been considered is entered (e.g. Cyrillic characters or no characters at all), the program does not accept the input password and an additional popup is displayed, informing the user to try again with another one (Fig. 2). This will keep on happening unless the user enters a valid password, after which the password is stored as usual.

Following all this, the algorithm finally begins its work with its main part. First it tries to guess the password using the faster of the two methods, that being the dictionary approach. Mimicking the operating principle of a dictionary attack, it opens and reads through a text file containing an extremely long list of 10 million common passwords. Credit goes to GitHub user *danielmiessler* for providing free access to this list [7]. For each password on the list the program checks whether it matches the actual password that was received from the user. If a match is found, then the algorithm automatically stops running and a separate text file is opened in a new window. This file solely exists to serve a ”cosmetic” purpose as it only simulates what personal data might be stored inside a database of user accounts (Fig. 3). Otherwise, if the entire password list is checked and the password still remains to be found, the brute-force method is initialized.

In most cases, the operating principle of the brute-force method has slower speed than that of the dictionary approach but given enough time it is guaranteed to find the exact password. When it is initialized, it uses the previously created special list of characters, the length of the input password and the *itertools* Python library to iterate through all possible combinations of that length and with characters from that list (Fig. 4). When the password is found, the algorithm process is automatically terminated and the aforementioned ”cosmetic” text file is opened in a new window.

The GUI is also fully developed in Python but additionally uses the features of the PySimpleGUI package as a means for its implementation. It consists of 2 distinct windows (Fig. 5, Fig 6.), each integrating a certain functionality from the described password-cracking algorithm above. In accordance to the syntactical and semantic rules of the PySimpleGUI library, each of them is constructed using a standard layout (Fig. 7) and its runtime is regulated in a *while* loop. The first window that the user can interact with, lets them input a password in an input field. This window is integrated to work with the code described for the password input and storage and the character list that follows. Upon valid submission, the password is stored in the memory and the special list of characters is created for a potential brute-force method, after which the window is closed (Fig. 8). Following this, the second window is automatically opened, consisting of a stopwatch timer, a displayed time limit and an action button for simultaneously manipulating the stopwatch and the main part of the algorithm. Using this button, the user is able to:

- run the algorithm and start the stopwatch;
- prematurely terminate the running algorithm and stop the running stopwatch;
- reset all variable values in the algorithm and reset the stopwatch.

The code for this specific window is relatively long but this is due to the fact that a lot of things need to be updated whenever some event occurs. For instance, while the time is running, it is updated every 0.01 seconds. With PySimpleGUI there is a specific way to achieve this, which uses the *time* Python module to calculate the current time dynamically while also displaying it in the format MM:SS:MM (minutes:seconds:milliseconds). There are 3 cases of events that could happen during the runtime of the timer. In the first case, the timer stops running automatically once the program finds the input password and a success status is displayed below the timer, informing the user that the password has indeed been cracked (Fig. 9). In the second case, if the timer naturally exceeds the displayed time limit, the algorithm stops its work and the program shows a failure status (Fig. 10). This means that the algorithm has not been able to find the user's password within the time limit. In the final case, if the action button is clicked prematurely, the algorithm is terminated and again an appropriate message, reflecting its termination status, is displayed (Fig. 11). In all cases, the action button is also reconfigured to work as a reset button for the window. If it is clicked again, then the window's default values are restored, allowing the user to run the algorithm once more if they wish so but with the same password. Speaking of the default values, the default state of the window can also be seen the first time it is opened, i.e. after entering a valid password in the first window. In this state the time is set to 00:00:00, the values needed for calculating the time are all set to 0 and the action button is configured to simultaneously start the timer and run the algorithm.

It is also noteworthy to mention how the algorithm is implemented to work with the second window. Whenever it is initiated (with a click of the action button), the dictionary method is ran first (Fig. 12). Despite the size of the text file that is used, Python's built-in way of iterating through each password string is extremely fast and efficient. This makes the search for the password a very quick task than can be done in under 1 second, which essentially minimizes the margin of error that can occur in the measurement of

the dictionary method's time and thus the timer is allowed to begin running after this delay. However, that is not the case with the brute-force method. It is initialized only if the password is not found by the dictionary method. Since in most cases it can take substantially more time to find the password than the dictionary approach, both it and the timer need to run at the same time. This is achieved using the *threading* Python module, which allows this part of the algorithm to be ran in a separate thread independently, while leaving the timer run in the main one (Fig. 13). As a result, the program should theoretically smoothly display the running timer in the foreground (the window) while simultaneously working on cracking the password in the background. In practice, however, this implementation does not work as expected, which is to be discussed in the following "Assessment" section.

6.4. Assessment

The goal of the technical deliverable was to provide a program simulating a password-cracking process that could show users how vulnerable their passwords are to hacker attacks. This has been achieved with the help of the Python programming language utilizing the language's built-in tools to develop a back-end algorithm and the PySimpleGUI library to create a suitable graphical user interface (GUI) for the front-end.

Most of the defined requirements for the program have been fulfilled, including the simplistic design and the integration of the algorithm with the GUI. However, there remains one unsolved problem with it - the proper implementation of brute-force method. As previously mentioned, the expected result was to have the timer and the brute-force method run independently at the same time. In reality, all attempts to fully realize this with the use of threading have been unsuccessful. In the produced program, whenever the brute-force method is initialized in a separate thread, the timer continues to exist in the main one but is never updated, and thus remains in its default state (00:00:00) at all times. This means that while the algorithm is able to do its work properly in the background, it can never be automatically stopped by the 5-minute time limit, since the timer can not actually reach it. Moreover, the brute-force method appears to become too dependent on processing power after about 10 seconds from its initialization. As a result, the program becomes unresponsive, making it impossible to prematurely terminate the process using the "Stop" button for passwords longer than 5 or 6 characters. These issues seem to originate from the threading implementation of the brute-force method, since previous testing of the standalone timer and its integration with the dictionary method has not been problematic.

Overall, the technical deliverable manages to reach some of the main expected results, such as a simple-looking and easy to use GUI and a functional password-cracking simulation algorithm. Despite this, there are a handful of problems that have arisen and are yet to be solved, specifically the proper implementation of the brute-force method and the issues that it brings, most notably failure to run the timer concurrently and the heavy load on the processor. It is very likely that the author's lack of experience with threading in general and his lack of success to fully explore the PySimpleGUI library are the cause of these problems. Nevertheless, to some extent the produced program is still capable of showing the user how vulnerable their password is to malicious hacker attacks.

Acknowledgment

The author would like to thank the BiCS management and education team for the support and the amazing work done.

7. Conclusion

To sum up, a large portion of the total defined goals and objectives has been completed. Therefore, the project can be considered as successful for the most part.

With the in-depth level of information that this project has provided in regard to investigating the impact of passwords in online security, it is very much expected that readers of this report have become more aware of their own password creation and management practices and that they will look to improve them in the future.

The project is certainly not without its share of failures, most notably the lack of realization of some of the technical deliverable's functionality. The technical issues can absolutely be fixed but that will be achievable once the author has gained the necessary knowledge and experience to properly tackle the problem. After the fulfillment of this task and some further optimizations, the program will have the potential to be used as a powerful tool for large-scale campaigns that focus on revealing the truth to online users about how vulnerable to malicious attacks their passwords really are.

References

[1] Viktor Taneski, Marjan Heričko, Boštjan Brumen, "Systematic Overview of Password Security Problems", Acta Polytechnica Hungarica, Vol. 16, No. 3, p. 144, 2019

[2] Katha Chanda,"Password Security: An Analysis of Password Strengths and Vulnerabilities", International Journal of Computer Network and Information Security(IJCNIS), Vol.8, No.7, p. 23, 2016.DOI: 10.5815/ijcnis.2016.07.04

[3] Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Rich Shay, Tim Vidas Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Julio Lopez, "Guess Again (and Again and Again): Measuring Password Strength by Simulating Password-Cracking Algorithms", CMU-CyLab-11-008, p. 1, August 31, 2011

[4] Blase Ur, Jonathan Bees, Sean M. Segreti, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, "Do Users' Perceptions of Password Security Match Reality?", p. 3748, CHI'16, May 07-12, 2016, San Jose, CA, USA

[5] Elda Kuka, Prof. Assoc. Dr. Rovena Bahiti, "Information Security Management: Password Security Issues", Academic Journal of Interdisciplinary Studies, Vol 7 No 2, p. 43, July 2018

[6] Shannon Riley, "Password Security: What Users Know and What They Actually Do", Usability News, Vol. 8 Issue 1, February 2006

[7] Link to 10 million password list on GitHub: https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-1000000.txt

8. Appendix

```
lowercase = "abcdefghijklmnopqrstuvwxyz"
uppercase = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
digits = "0123456789"
special = "`~!@#$%^&№€*()_+,./\<>?|-="
```

Fig. 1. All allowed characters for a valid password

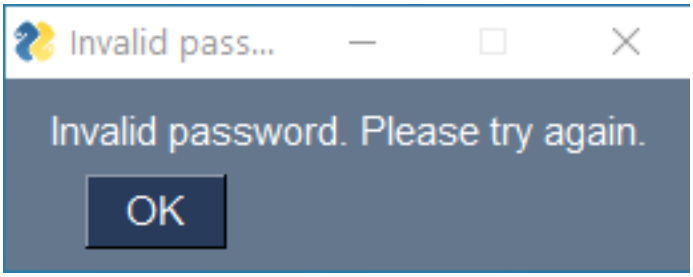

Fig. 2. Invalid password popup

```
First name: John
Last name: Smith
Email: johnsmith@mail.com
Phone number: +352 111 222 333
Date of birth: 01/01/2000
Country: Luxembourg
City: Luxembourg City
Street: Real Street 21
National identification number: 0123456789
ID card number: 9876543210
Bank account information:
- account holder: JOHN SMITH
- card number: 1234 5678 9876 5432
- PIN: 1234
```

Fig. 3. Sample data contained in ”cosmetic” file

```
def combinations(l,p):
        return itertools.product(l,repeat=len(p))
```

Fig. 4. Function returning all possible combinations for a password ”p” with length ”l”

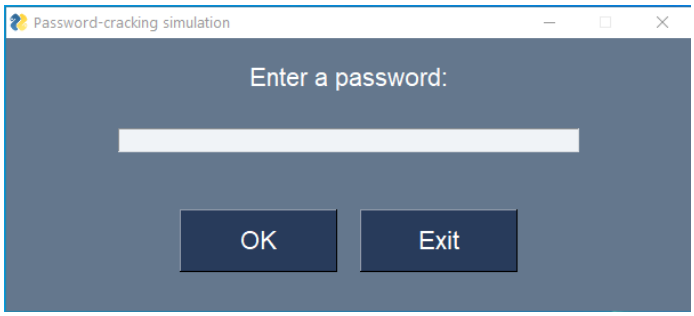

Fig. 5. Program window 1

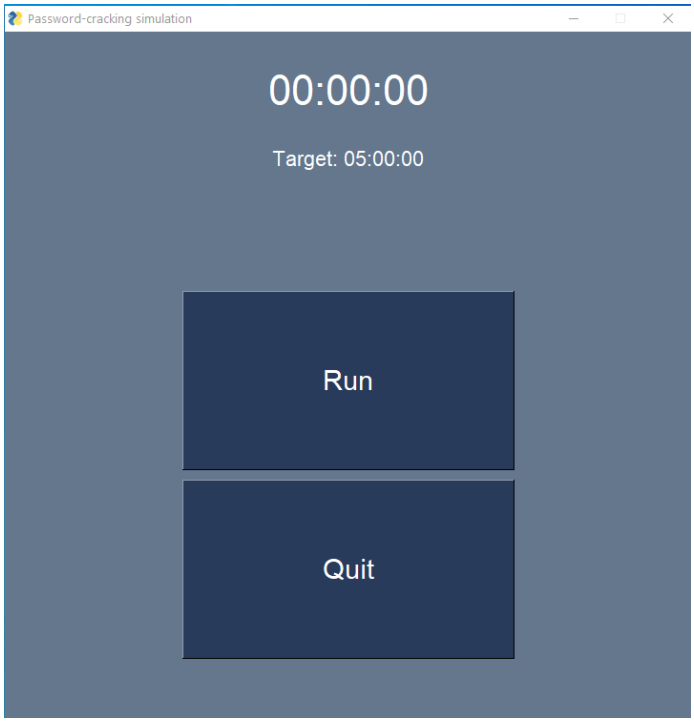

Fig. 6. Default state of program window 2

```
passWindowLayout = [
    [sg.Text("Enter a password:", pad=((0,0),(15,30)),font=("Helvetica",15))],
    [sg.Input(size=(50,1), pad=((0,0),(0,20)), enable_events=True, key="INPUT")],
    [sg.Button("OK", size=(10,2), pad=(10,30), font=("Helvetica",15)), sg.Button("Exit", size=(10,2), pad=(10,30), font=("Helvetica",15))]
]

attemptWindowLayout = [
    [sg.Text(size=(20,1), pad=((0,0),(30,0)), font=("Helvetica",30), justification="center", key="TIMER")],
    [sg.Text("Target: 05:00:00", font=("Helvetica",15), pad=(0,30), key="TARGET")],
    [sg.Text(size=(20,2), font=("Helvetica",25), justification="center", key="OUTPUT")],
    [sg.Button("Run", size=(20,5), pad=((0,0),(0,5)), font=("Helvetica",20), key="ACTION")],
    [sg.Button("Quit", size=(20,5), pad=(5), font=("Helvetica",20) )]
]

passWindow = sg.Window("Password-cracking simulation", passWindowLayout, size = (600,240), element_justification="c")
attemptWindow = sg.Window("Password-cracking simulation", attemptWindowLayout, size=(720,720), element_justification="c")
```

Fig. 7. Code for the layouts of both windows

```
while True:
    passWindowEvent, passWindowValues = passWindow.read()

    password = passWindow["INPUT"].get()

    if " " in password:
        passWindow["INPUT"].update(password[:-1])

    if passWindowEvent in (sg.WIN_CLOSED, "Exit"):
        break

    if passWindowEvent == "OK":
        if len(password)!=0 and isValid(password, lowercase, uppercase, digits, special):
            charsList = list(createCharslist(password, lowercase, uppercase, digits, special, chars))
            length = len(password)
            passWindow.close()
```

Fig. 8. Code that runs the first window

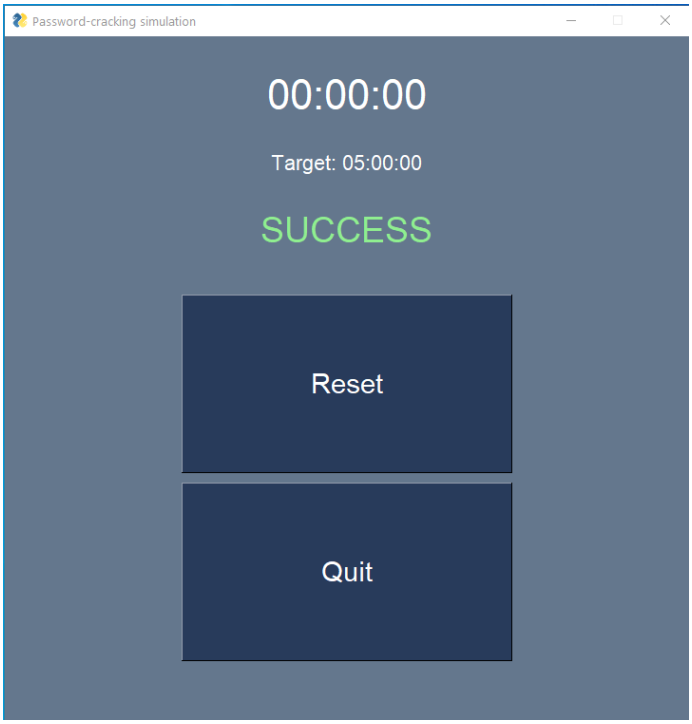

Fig. 9. State of window 2 after successfully finding the password

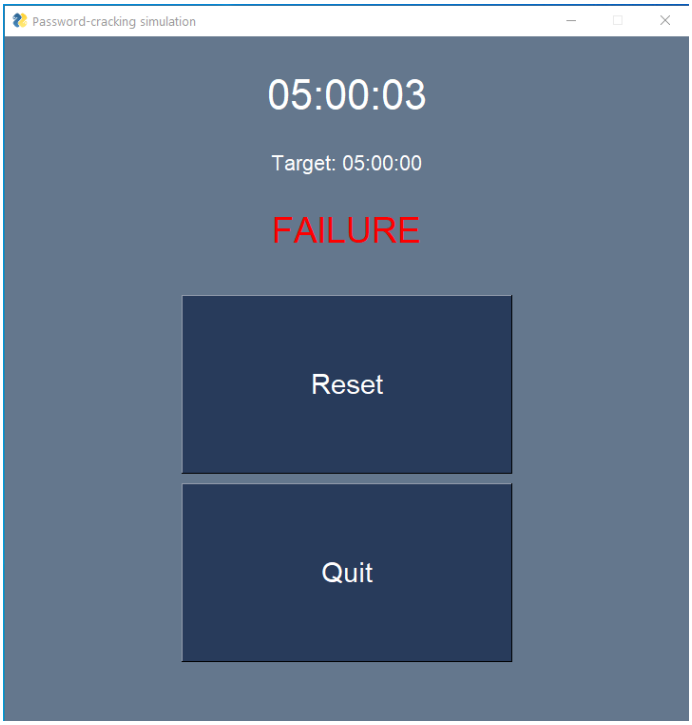

Fig. 10. State of window 2 after failing to find the password in time

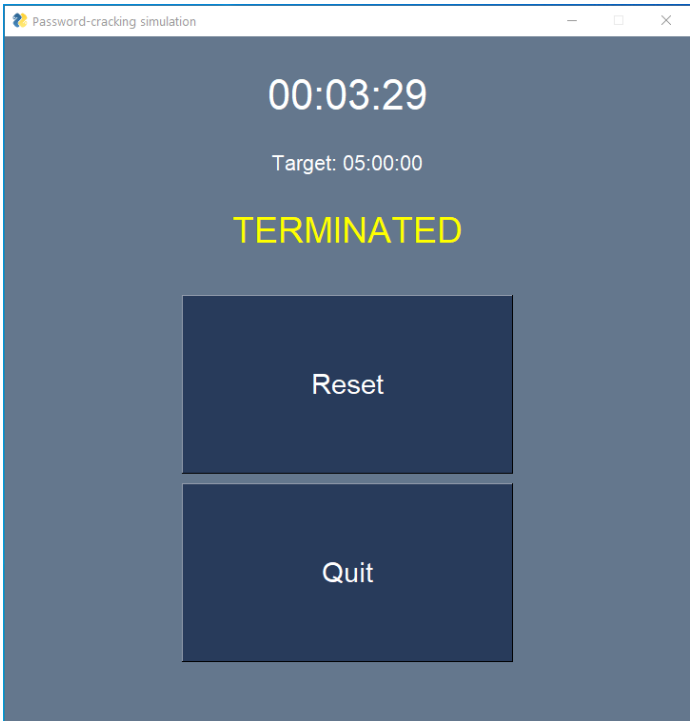

Fig. 11. State of window 2 after prematurely terminating the process

```
if guess != password:
    if guessPassword1(password):
        paused = True
        pausedTime = timeInt()
        attemptWindow["ACTION"].update("Reset")
        attemptWindow["OUTPUT"].update("SUCCESS", text_color = "lightgreen")
        process = subprocess.Popen(["important_info.txt"], shell=True)
        process.wait()
```

Fig. 12. Integration of dictionary method with GUI

```
else:
    if not thread_running:
        t = threading.Thread(target = guessPassword2, args=[guess,password]).start()
        thread_running = True
if password_found:
    paused = True
    pausedTime = timeInt()
    attemptWindow["ACTION"].update("Reset")
    attemptWindow["OUTPUT"].update("SUCCESS", text_color = "lightgreen")
    process = subprocess.Popen(["important_info.txt"], shell=True)
    process.wait()
    guess = ""
```

Fig. 13. Integration of brute-force method with GUI (only partially successful)