



Roskilde University

MASTER THESIS IN COMPUTER SCIENCE/INFORMATICS

# **A study on how to simplify strong password creation without sacrificing security**

*author:* Dragos ILIE

*supervisor:* Benedicte FLERON

4<sup>th</sup> September, 2020

# **ABSTRACT**

Within this thesis, I analyzed the password creation step that arises when users try to create login credentials on an online platform, in order to find out a way to greatly simplify said creation without reducing any aspect of its security potency. The thesis started off with a recap of the most known security challenges that impose the creation of strong passwords. The theories could be split into two perspectives and were mostly part of the Security field, as well as the User Experience one, and they were used to demonstrate why there is still room for improvement for the concepts and methods used when it comes to authentication.

Existing solutions on the market were summarized, and a prototype was created to illustrate another approach that has been thought of as a solution for the nuisance of having to come up with and remember strong passwords (and thus violating user experience values).

As the prototype was later validated through comparison to the existing solutions, the result was successful into showing that the problem formulation can indeed be answered, and it offered a practical display on how it can be achieved.

# TABLE OF CONTENTS

- 1. INTRODUCTION..... 1**
- 1.1. PROBLEM INTRODUCTION..... 1
- 1.2. PROBLEM FORMULATION..... 1
- 1.3. RESEARCH QUESTIONS..... 1
- 1.4. THESIS LIMITATIONS..... 2
  
- 2. THEORY..... 2**
- 2.1. AUTHENTICATION..... 3
- 2.1.1. Login system..... 3
- 2.2. ATTACKS AGAINST PASSWORDS..... 4
- 2.3. ENCRYPTION..... 8
- 2.4. STRONG PASSWORDS, ROLE AND NECESSITY..... 10
- 2.5. USABILITY IMPLICATIONS..... 13
- 2.6. EXISTING PASSWORD SOLUTIONS..... 15
  
- 3. PROTOTYPE..... 16**
  
- 4. VALIDATION OF PROTOTYPE..... 20**
  
- 5. DISCUSSION..... 22**
  
- 6. CONCLUSION..... 23**
  
- 7. REFERENCES..... 23**

## 1.1. PROBLEM INTRODUCTION

Passwords or some sort of primitive form of authentication have been present in the life of humans since the ancient times. They have always been used to confirm the identity of the individual in order to give him access to secret or protected information, belongings, rooms, gatherings, etc. that were not available to the regular citizens. Thus, from the ancient civilizations, people have always felt the need to protect various attributes for the sake of safekeeping, which makes it safe to say it is only natural that in the digital world passwords be present in one way or another (Wikipedia 2020 – Password). But does a password provide the ultimate protection to its owner against prying eyes? What happens when the password is overheard or intercepted by third parties? Or even when it is too basic or common and it can simply be guessed?

In the IT world, the field of cyber-security is trying to provide a solution to these issues: there are many cases in which corporate employees are following introductory courses for security regulations in order to learn basic protection of their accounts and, implicitly, the company's property; they are taught not to write the passwords in unprotected documents or on pieces of paper, they are taught not to click dubious links or disclose their passwords to anyone, etc. (Usecure.io 2020). One of the approaches that has become a rule in the recent years is to force the users to choose strong passwords, complicated strings that are meant to discourage or greatly delay any theft attempt; they achieve this by being much harder to guess by regular attacking tools that informational thieves use. However, as the field of Usability is rapidly gaining ground and all the tedious workload is being shifted towards the machines for the sake of user-friendliness, why cannot machines be forced to accept the weak passwords from users and edit them into strong ones? Having this concept in mind, I conducted a study presented in this thesis in order to find out if this shift mentioned above is a viable solution when it comes to creating passwords.

## 1.2. PROBLEM FORMULATION

***How can we shift the burden of creating and thus remembering strong passwords from the users to the authentication system, without sacrificing security?***

## 1.3. RESEARCH QUESTIONS

- Why do websites impose the use of strong passwords when creating an account?
- How are the strong passwords stored by the system, having security considerations in mind?
- How successful are the solutions that already exist for password management?
- How can we reprogram an authentication system to generate content so that a weak password is edited and stored as a strong password?
- Can the reprogramming be done while retaining all the security invulnerabilities and without hindering the user in any way?

This thesis will analyze the treatment that passwords receive from their inception to their storing, as well as authentication, and wants to provide a prototype that will prove the idea behind the problem formulation, that systems can take the weight of creating strong passwords instead of the users. This thesis does cover neither the strategy of launching said prototype on the market nor the actual launch, its aim is to show a successful version of a prototype, in a controlled environment. It does not contain an extensive session of security testing and the scenarios in which a hacker could launch an attack on it have not all been exhausted. The prototype does not treat, nor protect against other security flaws, such as SQL injection, XSS attacks, DOS attacks, etc.; it strictly tries to provide an example of a solution for the problem formulation. The design is not a defining priority in the creation of the prototype either.

When analyzing how the prototype could perform once integrated into a real-life system, it is assumed that said system already contains modern security considerations, so as not to rule out the prototype as faulty because of aspects that do not deal directly with passwords.

## 2. THEORY

Since the beginning of the internet, security developers have been engaged in a race around the clock with the virtual information thieves (commonly known as hackers). They would be responsible with testing the existing systems to make sure that they are, if not impossible, extremely difficult and resource-consuming to penetrate, in order to discourage any enthusiasts from trying to seize control over sensitive data. Along with the evolution of the IT environment, gadgets and platforms, and for various other reasons (user-friendliness, applicability into other fields – army, finance, healthcare etc.), newer and more sophisticated technologies have risen in order to provide functionality that in the past would have been placed within the Science-Fiction realm (Augusto, J. 2010). However, as one can imagine, this rise in sophistication also made way for a lot more opportunities of exploitation than before, which in turn made the job of aforementioned security experts more crucial than ever.

Nowadays, within the field of cyber-security, which deals with making a system as impenetrable as possible, 3 factors have been identified and derived as primary concerns when it comes to designing a system: Confidentiality, Integrity, Availability (abbreviated C.I.A.); besides these, the other factors are considered secondary (Stallings, W., & Brown, L. 2018).

**Confidentiality** – the characteristic of a system to allow the protection of data considered sensitive by its owner, as well as protecting the identity and privacy of users on the world wide web;

**Integrity** – the characteristic of a system that ensures the data that needs to be accessed by a user has not been tampered with/modified in any way, in order to spread misinformation and alter choice outcomes amongst users;

**Availability** – the characteristic of a system that ensures the data is accessible to the

user at any time of choice and from any geolocation and platform, within the limits set by the existing technology (Stallings, W., & Brown, L. 2018).

Following these accepted security policies, passwords play a crucial and practical role into protecting the data that is considered sensitive by the various parties connected through the internet and, thus, one can say they contribute to implementing a lot of the security policies presented above. Since studies about passwords show that they are here to stay, with no viable replacement options found yet (Herley, C., & Van Oorschot, P. 2012), the developers must try to keep them up to date with the latest trends and discoveries in the hacking world and patch the password usage according to the most recent vulnerabilities found.

## 2.1. AUTHENTICATION

According to Stallings, W., & Brown, L. (2018) and Wordfence (2020), the means of authentication can be split into four main categories:

- “Something the individual knows” – passwords, PIN codes, or even answers to preset questions;
- “Something the individual has” – these possessions are named tokens and they can be both physical (e.g. smart cards) and virtual/digital (private keys, e.g. the digital passport);
- “Something the individual is” – facial, fingertips, retinal scan recognition systems;
- “Something the individual does” – voice, handwriting, walking pattern recognition systems

These elements, when properly implemented, they provide a certain degree of security. However, all of them have flaws: a password can be hacked, a key card can be stolen, etc. In order to counteract these problems, multifactor authentication has been created, and it can consist of implementations of 2 or more of the categories above. For instance, the Danish NEMID system is a 2-factor authentication method (password and physical token), internet banking is a 2-factor authentication (password and SMS systems).

However, as this thesis focuses on studying the passwords, the analysis disregards the other categories and focuses on “something the individual knows”, mainly passwords.

### 2.1.1. Login system

From a strictly technical point of view and through a very simplified explanation, an online platform with a credentials system in place works by following a certain number of steps, which, amongst others, could be looked at from 2 different perspectives, Identification and authentication. (Stallings, W., & Brown, L. 2018; Adams, A., & Sasse, M. 1999).

**Identification:** the system uses the input from the user, which it adds in the database under the table allocated for that purpose. The input fields may vary depending on what the owner of the system wishes to register about its users, but it must contain, in one form or another, a username and a password. Changing passwords can be considered a

subcategory of this process, because whenever the user decides to update his passcode, the system follows the same steps as during the creation (the only difference is replacing the value in the database instead of adding, with the preceding and the antecedent steps being the same).

**Authentication:** after the account has been set up, whenever the user needs to log in, he needs to provide the matching credentials in order to be identified as the owner of the account. The system then takes the input given (in most cases a username and password) and it compares it to the data from the database. If they match 100%, then this means the user is genuine and is therefore granted access to the account information.

Even though the system presented above is simplistic and of stock version (no security implementations for passwords), it is safe to assume that when the login function had been invented, the systems in place were more or less resembling the aforementioned (Wordfence 2020). What determined the change along with the evolution of the systems and their security, and what the improvements were, will be concluded at the end of subchapter 2.2. *Attacks against passwords*.

## 2.2. ATTACKS AGAINST PASSWORDS

As explained above, a lot of the companies keep sensitive information in databases, stored on the online servers of their Internet Service Providers and Hosting Providers. This information is crucial for one's business, as it contains data that can cause a disaster for the company if it leaks, especially in the case of corporations that are responsible for protecting millions of usernames and passwords (CSOonline 2020). A contributing factor to this kind of disaster is the use of master passwords, which will be explained later on in this thesis.

(CSOonline 2020) shows through Figure 1 the most impactful breaches of the 21st century. As it can be noticed, some important and famous brands/corporations in various markets have been affected by the hackers, which only goes to show that no business' system is safe against threats if mismanaged, no matter the prestige or popularity (Loo, A. 2008). An additional material supporting the claim of hacked businesses and acting as an updated list is the one provided by (Informationisbeautiful 2020). They present all the breaches that have had more than 30.000 victims, starting from 2009.

It is also curious to note that the data available spans from the 2000s and forward, which implies that, before, the online platforms were not as spread, as sophisticated, or purely the turnover was not attractive enough to actually be worth a deployment of resources for a mass break-in (in these cases millions/hundreds of millions/billions of accounts had been compromised at a time). An interesting aspect that is also worthy of noting is that the attackers are of various backgrounds, which shows that they cannot be categorized under one typology or another.

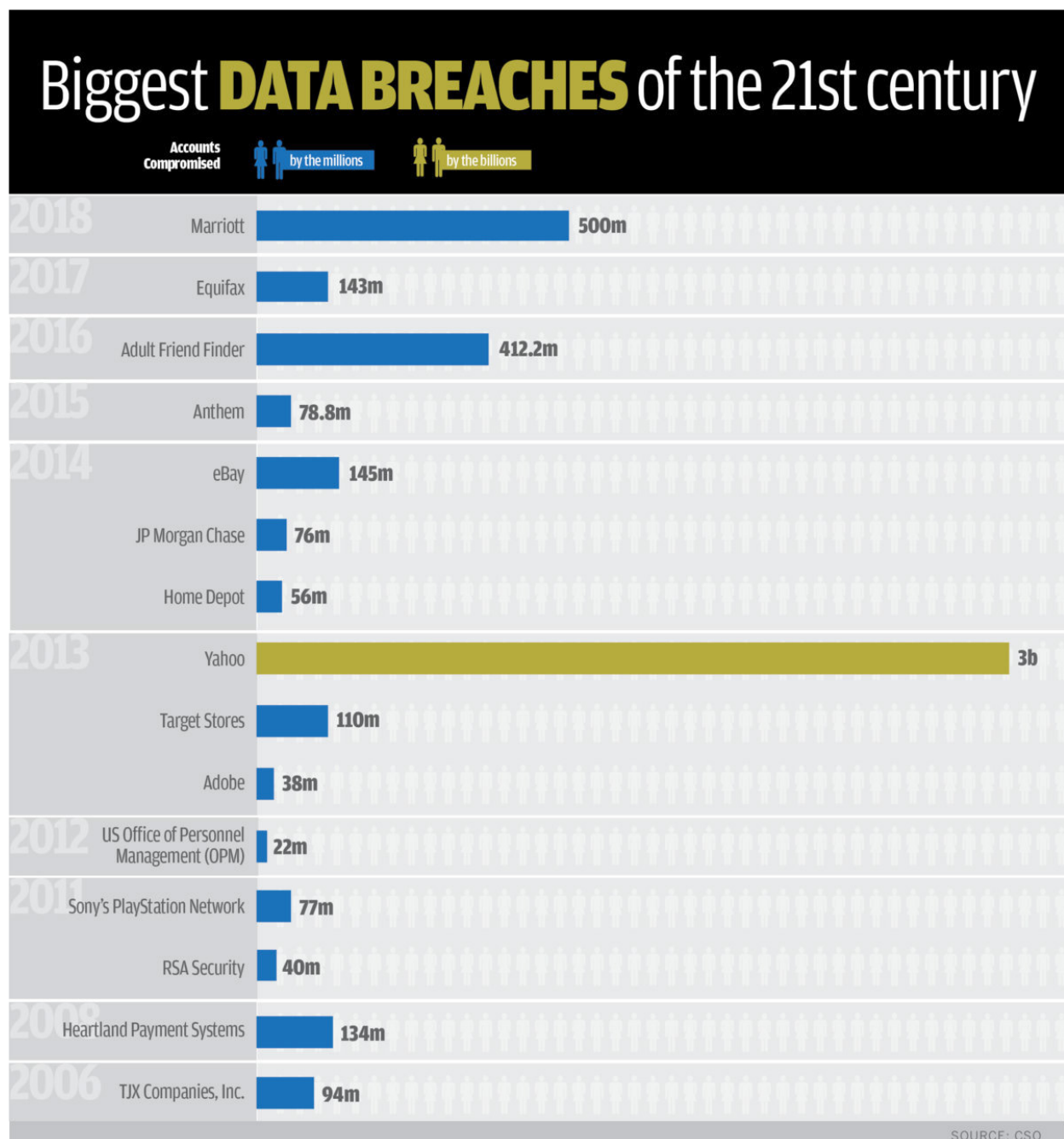


Figure 1 – Biggest data breaches of the 21st century, taken from csoonline.com

As (CSOonline 2020) presents its data, there are a few conclusions that can be drawn:

- all attacks were successful, putting at risk all the 3 security considerations:
  - Confidentiality** – people had their accounts compromised and data stolen;
  - Integrity** – the accounts' data was no longer reliable, could have been tempered with;
  - Availability** – since the accounts were broken, that means the perpetrator could deny access to the actual user;
- in all attacks, the people's identity was compromised, some had their financial



credentials discovered, and in 12 out of 15 cases, the password was present as stolen data, which only proves it is one of the preferred targets of the perpetrators, being sold on the black market along with the username. In addition to being sold for the benefit of penetrating a certain user's privacy and account, the passwords contributed statistically to the improvement of the attacking tools by creating a precedent, a list that can be used as a starting point for future attacks. Therefore, confronted with all these factors, it is safe to assume that since the passwords offer so many advantages, they need to be very well guarded.

There are many reasons why a leak can happen: a hacker penetrating the system, a disgruntled employee with access to the database, physical hardware stolen, mismanagement of the sensitive data within the company, badly planned risk mitigation, not allocating enough resources to protect the important information, etc. As Stallings, W. and Brown, L. (2018) tell us in their well-written book, there are many types of attacks that can hit a system, some of which are directly focused on stealing passwords from databases. I present some of the most popular below.

**“Offline dictionary attacks”** – also known as brute-force, these types of attacks focus more or less on systematically guessing the password from a database. Hackers find a way to infiltrate into the system by bypassing the security measures and they end up getting ahold of the system password file. Countermeasures to this kind of attack impose a tight control/preventing unauthorized access to the password file, and upon intrusion detection, passwords must be reissued rapidly to mitigate the risk. To put this in perspective, If the passwords were to be stored in plain text, as hinted under subchapter **2.1.1 Login system**, the hackers would use the username/password combination to gain access. However, later in this thesis I will explain how encryption stops the hackers in this particular situation.

**“Specific account attack”** – with this type of attack, the perpetrator supposedly knows beforehand or has an easy time in guessing a specific username, then submits random passwords until he succeeds in gaining access. A certain degree of acquaintance between the victim and the attacker is useful, but not mandatory. A typical countermeasure is locking the account after too many failed attempts or preventing further tries for a specific amount of time. This countermeasure is also used as a defense mechanism against denial of service attacks, since issuing too many iterations of a login procedure on a system without a safety measure may lead to overloading and incapacitating said system.

**“Popular password attack”** – can be considered a subset of the previous attacks and consists of using popular passwords against a wide range of usernames. As users tend to pick recognizable passwords (such as names of objects or persons), which in turn makes the password easy to guess. To countermeasure this, systems could be put in place that recognize the easily guessable passwords and force the user to pick something more uncommon.

**“Password guessing against a specific user”** – as mentioned above, the perpetrator can be familiar with the victim or he must gather knowledge about the victim's habits, preferences, names, etc. This specific case leads to an attack where the hacker can use

inside knowledge to gain the advantage; the knowledge can be either of the user or of the system in place. Countermeasures for this include training the users into choosing more difficult passwords and they are reinforced through applying certain triggers to the system that stop the choices that are considered poor: passwords must be different than usernames, they must contain lowercase and uppercase letters, numbers, special characters, must have a specific length, etc. (The ones that fulfill all these terms are known as strong passwords, which will be presented later).

**“Exploiting user mistakes”** – as explained above, strong passwords make for very good repellants against attacks on one’s account. However, a downside of having to deal with a strong password is the inclination of one’s owner to write it down, so as to make it easier to remember. This immediately brings the risk of theft to a much higher possibility, as the attacker can read the written password. The owner can also communicate the password to other colleagues for sharing accounts, which in turn leaves the exchange susceptible to eavesdropping. In addition, social engineering can be used, through various methods (phishing, spear phishing, baiting, malware, etc.) to trick the users into revealing their passwords to unknown parties by resorting to emotionally appealing the human intellect (Erbschloe, M. 2020). Countermeasures to these threats include carrying out internal trainings with the human factor of the equation in which they are taught the dangers of these actions and why they are considered reckless.



Figure 2 – Password leak, taken from arstechnica.com

Figure 2 demonstrates hands down the dangers that the writing of passwords poses. According to (Arstechnica 2020), during an interview of an employee from a French television network, viewers were able to read a set of passwords belonging to the internal accounts of the company (they can be seen in the top-left corner in the image). This is a clear and applied example of the previous type of attack and why it should be avoided.

**“Exploiting multiple password use”** – also known as a master password, it will be analyzed later on. In this case, people who use the same password over different platform make it a lot easier for attackers to penetrate the different systems accounts of the same user. Countermeasures include forbidding users to pick similar passwords.

To conclude all the information presented above, there are certain barriers that can be posed between an attacker and the credentials' virtual location. However, if they were to be stored in their stock version (plain text) in the database, as presented in the subchapter **2.1.1 Login system**, it would mean the ideal scenario for the culprit, as he/she would not have to fight any other defense mechanism that concerns the actual password. So, in order to avoid this and to provide a better protection, the developers had created encryption protocols which, when used, would store a scrambled version of the original user input, rendering useless the skimming of the rows of the database by the human eye. (Stallings, W., & Brown, L. 2018).

### 2.3. ENCRYPTION

According to Wikipedia – Encryption (2020), encryption is the “process of encoding information” and it has existed in form or another since the ancient times. It has been used for both military applications and civilian, in the past, while nowadays, in the digital era, it is also helpful under cryptography patronage, for security over the internet for a plethora of fields (financial, medical, etc.). Its main use is to protect sensitive/private information that is sent over unsecure channels or, as follows, to protect online password storage. As the definition suggests, it deals with encoding information so that a malicious individual cannot get ahold of the information, while the receiver can decipher it successfully (Kessler, G. 2020).

However, for practical applications regarding password security, encryption translates as hashing and it started out as a protection against hackers which would, when getting access to a plain-text password file, download it and instantly gain access to all the accounts listed. The developers' answer to this threat was to create hashing algorithms, which would take any string given as an input and release an output string of a certain number of characters: for the sake of this explanation, let us assume 32 characters (Han et al., 2014).

The innovation that this method brings to the table is that no matter the length of the input, the output will contain the same number of characters; in addition, a string that will be submitted through the hashing algorithm will always produce the same output result, called hash. This, in turn, opens up the opportunity of being able to handle the hashed version of the password for any of the steps (creation of account, credentials change, user authentication), without even having to get in contact with the actual plain text sensitive information, providing the developers with the foundation for the fight against password theft (Wordfence 2020).

However, hashing algorithms are not flawless and the breakthrough that allowed them to be useful in the first place is the key to their weakness: if an input string has a unique corresponding output string regardless of the variables, then the solution to hacking these hashes is to use tools that try every possible combination of strings until their hash matches the one from the stolen database sector. This is a more appropriate description of the Brute-force attacks mentioned in **2.2. Attacks against passwords** and in actuality all it does is to greatly delay the attackers in finding out the passwords, time dependent on their hardware configuration and capabilities. Alternatively, since the hashes are always the same, a collection of the most popular string combinations (words

and such) and their hashes can be put in place, greatly reducing the time it takes to guess the password, since the process now excludes the time required to create the hashes and it simply skips to the comparisons. Such a collection of words is called a **Rainbow table** and it is a very convenient method to crack hashes with no additional security. In order to protect against these, the developers were forced to adopt salts in the password handling (Wordfence 2020; Han et al., 2014).

Since the rainbow tables contain hashes of known strings/words, the solution is to make said words adopt a random appearance. Therefore, simplistically put, a salt is a random generated piece of text that is allocated before or after the initial password, with the resulting combination being hashed afterwards. Since the salt is random, it renders the rainbow table useless. Since the result is still hashed, it greatly reduces the brute force attacks.

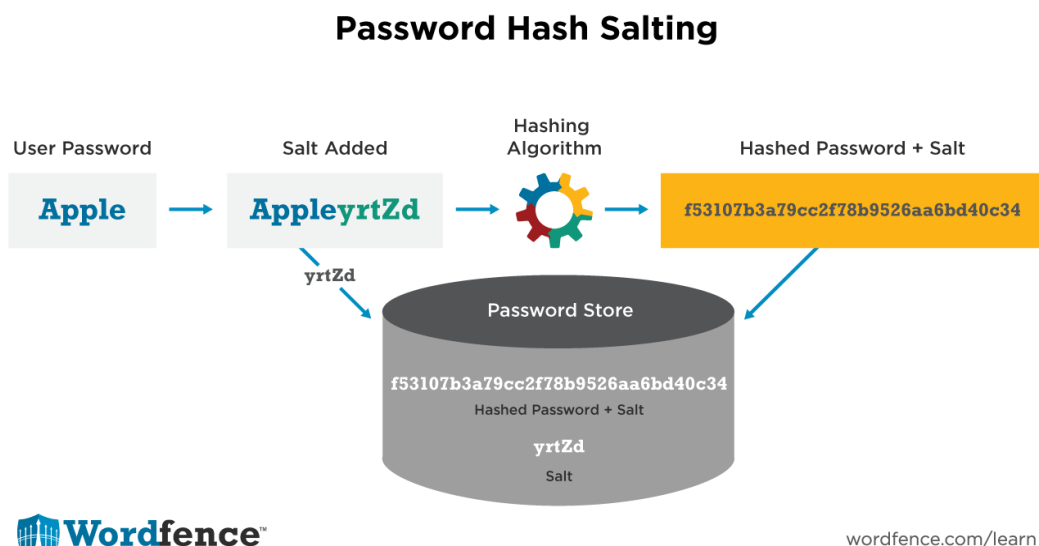


Figure 3 – Password hashing with salt, according to wordfence.com

Figure 3 displays a flowchart on how salting the passwords works. Therefore, using the newly gathered information on how encryption works, along with the examples from the above figure, the updated version of the authentication system presented at 2.1.1. **Login system** will have the following schematic representation (Wordfence 2020):

*When creating the account:*

1. A new user chooses the desired password, in this case “Apple”;
2. The system generates a random text, called salt, in this case “yrtZd”;
3. The system concatenates the password and the salt, in this case resulting “AppleyrtZd”;
4. The system uses the hashing algorithm on the combined string and ends up with a specific hash;
5. The system stores the hash from point 4, along with the salt in the specific database. It is important to note that if the salt were to be thrown away, the authentication would be impossible, since the salt is randomly generated.

*When authenticating:*

1. The same user as in the creation process wishes to login. He inserts his password, in this case “Apple”;
2. The system, knowing the username, retrieves the salt along with the hash from the database. Note that it is not the original password’s hash, but the original password + salt combination’s;
3. The system concatenates the inputted password with the retrieved salt, in this case “AppleyrtZd”;
4. The system hashes the string from point 3;
5. The system compares the string from point 4 to the hash retrieved from the database. If the match is positive, then the passwords are identical, so the user is valid and therefore granted access. A good observation is that this system does not even need to handle the plain text password at any point, which is what a good system should do, for a security and privacy point of view.

Because of this, the hacker’s attack strategy has to change from the previous approach. He now must concatenate the salt to any string he might attempt to submit. This renders the rainbow tablet useless and also hinders the brute force attack, since longer strings make up for longer permutations of characters, thus longer cracking time.

#### **2.4. STRONG PASSWORDS, ROLE AND NECESSITY**

As concluded by the previous subchapters, cracking a password cannot be avoided, but it can be delayed in such a way that the amount of time taken to crack it heavily outweighs the benefits, in order to discourage any eventual attacker.

Wordfence (2020) argues that since the hardware is evolving and the most modern GPUs can process string combinations at a much greater pace than when the salts were created, passwords need to be reinforced in some way. A password 9 characters long consisting of a combination of numbers and lowercase letters will give out 101,559,956,668,416 possible passwords (36 to the power of 9). Following suit, a password made out of 1 and 0, measuring 4 digits will give out 16 combinations (2 to the power of 4). Lastly, same 4-digit password consisting of lowercase, uppercase letters, numbers and 10 symbols will yield 26,873,856 combinations.

As Han et al., (2014) also present, the important conclusion that needs to be drawn from these calculations is that combinations of various characters from the keyboard, along with a healthy length of the password, can lead to improving the security in a radical manner. In addition, to back this claim from a research point of view, there are a lot of papers that study the importance of strong passwords and this only shows how important this actual topic is. However, as Lisa et al., (2010) tell us, “Computer security experts recommend the use of strong passwords, but do not necessarily agree on their exact characteristics. However, the majority of strong password recommendations do include the combination of letters, numbers, and symbols or special characters “. This shows that the experts slightly differ in approaches, which means that the field is under rapid development and no consensus has been reached yet. The same authors mention that “Last, users should change their passwords on a regular basis, with recommendations ranging anywhere from 30 to 120 days”, recommendation that is

completely overlooked in other studies of the same field. In their study, Komanduri et al., (2011) consider different levels of entropy (unpredictability) when studying the effect on password selection on users' behavior.

number of Characters	Numbers only	Upper or lower case letters	upper or lower case letters mixed	numbers, upper and lower case letters	numbers, upper and lower case letters, symbols
3	Instantly	Instantly	Instantly	Instantly	Instantly
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	3 secs	10 secs
6	Instantly	Instantly	8 secs	3 mins	13 mins
7	Instantly	Instantly	5 mins	3 hours	17 hours
8	Instantly	13 mins	3 hours	10 days	57 days
9	4 secs	6 hours	4 days	1 year	12 years
10	40 secs	6 days	169 days	106 years	928 years
11	6 mins	169 days	16 years	6k years	71k years
12	1 hour	12 years	600 years	108k years	5m years
13	11 hours	314 years	21k years	25m years	423m years
14	4 days	8k years	778k years	1bn years	5bn years
15	46 days	212k years	28m years	97bn years	2tn years
16	1 year	512m years	1bn years	6tn years	193tn years
17	12 years	143m years	36bn years	374tn years	14qd years
18	126 years	3bn years	1tn years	23qd years	1qt years

Key:

k – Thousand (1,000 or  $10^3$ )  
m – Million (1,000,000 or  $10^6$ )  
bn – Billion (1,000,000,000 or  $10^9$ )  
tn – Trillion (1,000,000,000,000 or  $10^{12}$ )  
qd – Quadrillion (1,000,000,000,000,000 or  $10^{15}$ )  
qt – Quintillion (1,000,000,000,000,000,000 or  $10^{18}$ )

Figure 4 – Password cracking time, taken from inetsolution.com

Figure 4, containing data gathered in 2012, shows the amount of time it would take computers from that year to crack passwords made up using different rules (Inetsolution 2020). To put the year factor into context, the table below presents three measurement tools from actual times which show how the hacking technique of brute-force attacks has improved over the years.

Password	Time to crack (Mylogin 2020)	Time to crack (Randomize 2020)	Time to crack (Online Domain Tools 2020)
“password”	Less than a second, the password is comprised of a very common word	1.:13 minutes	0 seconds
“P@s\$worD”	Less than a second, since the password is a very common combination of characters	24 days, 20 hours	Around 1 minute
“P@s\$worDasd”	3.46 seconds, since the password is a very common combination of characters, followed by a combination of characters close on the keyboard layout	57377 years, 10 months	Around 2 years
“CerP*%204”	12 centuries, very good combination of characters	6 years, 5 months	Around 2 hours
“earth is a planet like mars and jupiter”	10 thousand trillion years	6.725539236313682e+51 years	About 116 octillion years

The table above uncovers a very interesting and unique view. Naturally, according to the theory presented until now, a combination of uppercase, lowercase letters, numbers and special characters of more than 10–15 letters should be relatively safe against cracking. However, since the password cracking tools got improved, it is no longer considered safe practice to use the aforementioned combination. Because of habit, people tend to use passwords tricks that they can relate to (such as replacing “O/o” with “o”, “a” with “@”, etc.), or use letters that are close to the keyboard to make up for missing password length; more will be explained later in this thesis.

Therefore, as the second half examples show, using combinations that form no apparent word is beneficial. Coincidentally, the last example, which contains only lowercase English words and the “ “ special character is rated amongst the toughest to crack, even though, theoretically, it does not respect almost any of the rules. This could be called a contradiction, an exception to the rule, and it happens because apparently the extreme length of the password makes up for not adhering to almost any of the practices considered safe.

As a note, it can be deduced from the calculators above that there is no specific consensus between the security experts, no accepted rule that can always act as a guiding constant. However, before drawing any long-lasting conclusions, one must always consider the parameters that have been set when creating the calculators, like hardware and such other conditions (Ntantogian et al., 2019).

## 2.5. USABILITY IMPLICATIONS

As the explanations from previous subchapters prove, strong passwords are the way to go when discussing the protection that the encryption offers. However, as a lot of studies and surveys show, users are not always “on board” when it comes to the password regulations, and even after they have been presented with the risks (e.g. in introductory courses on security in corporations), some of them still fail to adhere to the rules. They disregard the protocols, they write the passwords down, communicate them to others, do not pick the best character combinations, etc. (Shay et al., 2010). According to Dell’Amico et al., (2010), people have tried finding a common ground between using and remembering strong passwords, which clearly demonstrates the hard time the users have when it comes to strong passwords, a clear violation towards user-friendliness. They give example of a practice in recent years where users create strong passwords by remembering the acronyms and punctuation of quotes: ‘For example, the phrase “Alas, poor Yorick! I knew him, Horatio” becomes “A,pY!Ikh,H”.’ However, this practice is being countered by hackers, who are trying to import collections of said quotes into their dictionaries.

For convenience purposes and to combat the strong password challenge, since the reflex is to avoid frustration, the user’s intellect has subconsciously devised some attempts to make the password selection easier: they would use known words and replace the needed characters within them (practice proved unsafe in 2.4. Strong passwords, role and necessity) or they would use master passwords.

A master password is essentially a password that is used in an unchanged form across multiple platforms; it can either be strong or weak, but it must be consistent in its format and it must unlock access for the same individual to multiple accounts (Wash et al., 2016). Since the credentials used in unlocking said accounts rely on a username and password, and not more than once, platforms ask for the email as a username, it is ok to assume that the practice of making use of master passwords is a bad approach. Since the backend of systems is hidden, for security purposes, a user can only trust that the developers have protected it against the latest security breaches and, what is more, the system is under constant update against the exploits that have not yet been discovered. This might be true for big corporations, but not so much in case of small businesses whose website the user is actively using.

Well, judging by the risks and procedures presented in this thesis, if an attacker gets ahold of a user’s credentials and the password is a master one, then it is risk-free to assume that the hacker will gain complete access over all accounts registered to the specific username/email address. If the concept of the master password is followed throughout its definition, then even the email address lead in the process will be accessible using the same password, which can rapidly turn into a costly loss of



information for the user. Therefore, with this explanation in mind, one can only assume that the use of master passwords is greatly discouraged because of security purposes.

On one side, this failure of compliance from the user makes the job of the developers much more tedious and their effort futile, and on the other side, it exposes their sensitive information to high risks, as some say that the path to breaching a system is still the human component from the equation, as it is considered the weakest link. Schneier, B. (2004) states ‘... security is only as good as its weakest link, and people are the weakest link in the chain.’; Loo, A. (2008) concurs “In most systems, the weakest components are the end users, particularly when they are accessing the corporation’s databases with wireless facilities at home.”

A good approach to understanding why this happens would be to look at it from another perspective: this bad habit is not embraced by security experts, but the users should not be labelled as “enemies” (Adams, A., & Sasse, M. 1999). From a cognitive point of view, the human memory is somehow limited when it comes to memorizing the overly complicated passwords (Wiedenbeck et al., 2005), especially since nowadays a lot of platforms use personal accounts. The passcode’s format is not helping, since the ones that are truly secure are composed of a string that has no meaning, contributing to the forgetfulness manifested by the user. If they were to respect the full security considerations, it would result in them forgetting the passwords that they rarely use or even the ones they use more often, in case of a very complex string of characters (Shay et al., 2010). In turn, this would lead to an emotional trigger which would make the users feel frustrated and annoyed. Therefore, it is safe to state that the matter of strong passwords turns into a Usability versus Security paradox: it is hard to uphold security without sacrificing usability and the other way around (Dell’Amico et al., 2010).

Researchers, though, have recognized the importance of incentivizing the user and this is why they have carried out studies about motivating them. Adams, A., & Sasse, M. (1999) identified that the users do not put a lot of effort into the security of their accounts and, even if they are aware of the security considerations, most of them lack motivation and they see authentication as a gate towards their end goals, a gate through which they should pass as fast as possible and without much stress. Some authors even composed a set of “recommendations” to help the security experts in addressing the users. Other researchers concluded that the reasons why people are not keen to respect the security policies are those of social and psychological nature (Sasse et al., 2001). However, newer studies (Yıldırım, M., & Mackie, I. 2019) show that the users can be persuaded to obey the security policies if the systems support them throughout the way: “Most systems that impose password restrictions offer their users password advice about creating passwords”. As such, there already are some solutions in place, that are not fault free though, as it will be shown in the next subchapter.

As Yıldırım, M., & Mackie, I. (2019) conclude, if the system works with the user instead of imposing a certain rule, the outcome is more likely to be favorable and people would listen to certain policies without that much struggle; they just need a motivating factor.

## 2.6. EXISTING PASSWORD SOLUTIONS

Thankfully, to address the challenge of users not respecting the password security guidelines, the IT developers have come up with a few fixes. In an attempt to appeal to the cognitive function of the users, some websites use passwords meters with colors representing weak (red), medium (orange) and strong (green) passwords. However, this meter is for advising purposes and it can be ignored by users. Other developers created algorithms that can allocate random passwords to the user, which are very safe indeed. The downfall of this approach, as explained in subchapter 2.2. *Attacks against passwords*, is that users will write this kind of password down, as it would be very inconvenient and not productive enough to remember. (Yıldırım, M., & Mackie, I. 2019)

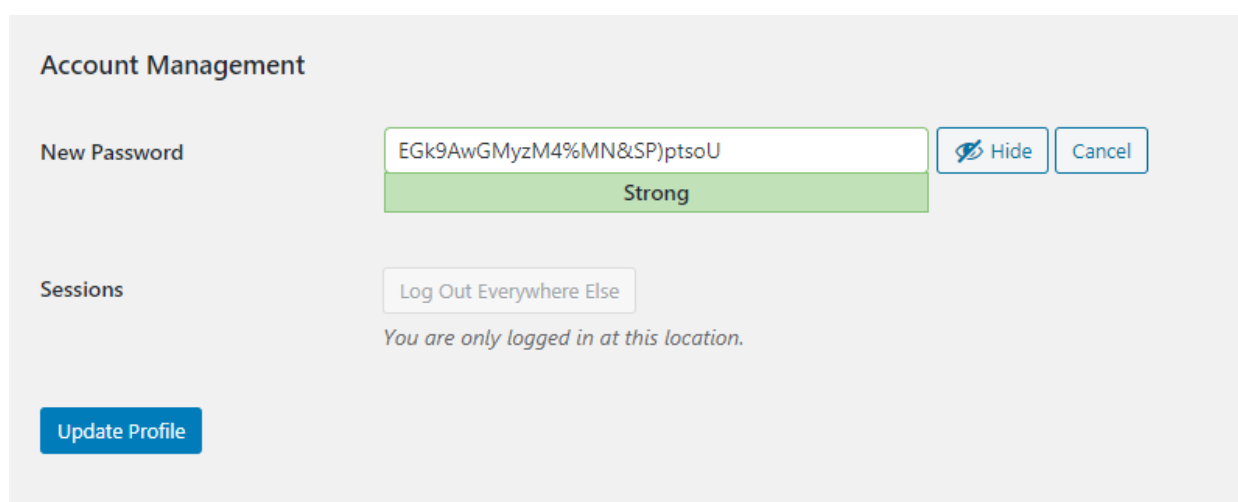


Figure 5 – Developer ideas to improve friendliness in password creation, taken from (Wordpress 2020)

Figure 5 contains the backend of the popular content management system Wordpress which illustrates the fixes presented above (both the random password generator and the password meter). The figure is a screenshot taken from the backend dashboard.

A third option that the developers went for is the creation of so-called password managers. According to Li et al., (2014), a password manager is a software whose purpose is to generate and store secure passwords for its users. The initial breakthrough is that a password manager works like a vault that can be open with a single password. Inside the vault there are all the passwords that the user needs for his various online accounts. Therefore, it takes away the cognitive limitation of the user by relieving him from having to create and remember difficult passwords. They can be built into the browser (e.g. Google Chrome™), standalone applications (e.g. LastPass™, DashLane™), or they can act as browser extensions (e.g. LastPass™).

Even though password managers could be viewed as improvements, studies show that users do not prefer them and that they do not come without flaws

Oesch, S., & Ruoti, S. (2019). published a paper which presents security vulnerabilities

that join the password managers (e.g. network injections). In addition, the way the passwords are stored presents flaws: they are decrypted for use with a key derived from the main password; since the main password must be stored in the database, this opens up ways of attacks as the ones mentioned under **2.2. Attacks against passwords**. If the main password were to fall to unwanted parties, then all the passwords that a user needs would be compromised. The risk of this is immense. Therefore, to track back to the general security considerations, it is beneficial to use a strong password. Which, in turn, given the negligent use of passwords explained in subchapter **2.5. Usability implications** can lead to mismanagement or even forgetting. It is easy to see why this becomes a vicious circle.

(Alkaldi, N., & Renaud, K. 2016) present a paper which deals with the usage and popularity of password managers (even though it focuses on smartphones, the point stands across devices). The survey carried out within it mentions that the password managers are not as widespread as it would be desired and gives as reasons: “Poor advertisement and a failure to reassure potential users about the trustworthiness of these applications could well explain the poor uptake of these tools. Moreover, the analysis reveals that designers should pay more attention to the user experience”. Ayyagari et al., (2019) conclude that “our study indicates that perceived vulnerability and perceived severity of password loss encourages the use of password managers. However, trust and ease of use resulted in counterintuitive findings”.

Based on all this research, it is clear that the existing solutions are, at best, mitigating the risk, not eliminating it, while password managers are far from perfect and they are not popular amongst users.

### 3. PROTOTYPE

This thesis and, by extension, this prototype, aims to explore the password creation process in the hopes of finding a solution for the existing problems which were presented in the previous chapter.

While examining the issues at hand, my idea was to force the burden of creating a strong password from the user’s mind to the login system, as it is clear that strong passwords are a hassle and are not exactly user friendly. So, the purpose was to alter a login system in a manner that no matter the password chosen by the user, the system would handle it in such a way that it would pass as a strong password in the database. Since the user did not know how the system was designed, a critical requirement was for him to be able to login with his chosen password, no matter how weak. Obviously, this kind of attempt came with some challenges of its own:

- For the solution to be viable, the password security policies must not have been inferior in any way to what had already existed;
- The missing text required for the password to pass as a strong one must have been created in a structural manner, so that it could be retrieved during the authentication phase (as explained in subchapter **2.2. Attacks against passwords**);
- The system must have considered best/worst case scenarios and have viable solutions for both.

### How the prototype works

In order to achieve a successful result, I had to keep in mind that the users would insert weak passwords as input, and I would have no way of knowing which input was missing in order to make it a strong password. I also needed to know that the generated text should be random. With these considerations, I turned my attention to the salt, which is a randomly generated string, that is kept in the database. The first step was to strip the salt of the letters and then use the string of digits resulted to generate the missing input for the weak password. Starting from backwards, I took the digits in groups of 2 and 3 and I turned them one at a time into the corresponding ASCII character (e.g. “a” in decimal ASCII code is “97”). The algorithm would then manipulate the string more than once, in order to accommodate for any of the missing characters, whichever they may be. (The parameters in this case at least were 1 number, 1 uppercase, 1 lowercase, 1 special character, with minimum 15 characters). After the characters were generated, they would be added to the back of the weak password. During authentication, the weak password would overgo the same treatment, since the salt is in the database and it would yield the same result, allowing the user to authenticate with a weak password.

It is important to note that since it is randomly generated, the minimum characters number could me bigger, and in case the salt would not be long enough, more mathematical operations could follow, like checking for the salt times 2, etc. Much more complicated equations can be added to the algorithm, ensuring that the hackers could not, in any way, find out the algorithm and thus facilitating their work. One option could be, for instance, incorporating the weak password’s ASCII codes to the algorithm (since that’s hidden, it would make for a very good defense). The separate document attached to this thesis will give an insight and a better understanding into how the code works.

## ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	.
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	-	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	.	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	:	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	;	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Figure 6 – The ASCII table, taken from google.com

## The interface

The solution that I came up with is a prototype written in HTML, PHP and MYSQL. The system encompasses a simple login environment accessible from a web browser. HTML is used to display the signup form, PHP is used to take the credentials, handle them in the background for both creation and authentication, and MYSQL is used to access the database in which the credentials are stored. The prototype uses the exact steps that were described under **2.2. Attacks against passwords**.

The image shows a web interface with two main sections on the left and a right-hand panel. The left section contains two forms: a 'Login' form with 'UserName\*' and 'Password\*' fields and a 'Login' button, and a 'Sign up' form with 'UserName\*' and 'Password\*' fields and a 'Sign up' button. Below these forms are navigation links: Home, Sign up, and Dashboard. The right-hand panel shows a 'Logged in as user6' status with a 'Logout' button, the same navigation links, a 'Sign up' heading, and a message: 'You cannot sign up, you are already logged in!!!'. At the bottom of the right panel is the footer: 'phone: 1234567890 © 2020 Dragos Ilie thesis'.

Figure 7 – The prototype's interface

Figure 7 shows the interface of the prototype. The main page consists of 2 simple forms that require a username and password (one for signing up, one for logging in) – they are displayed on the left. The right panel shows what the authenticated user sees: his credentials and a button allowing him to log out, as well as information only visible to the logged in users.

## The functionality

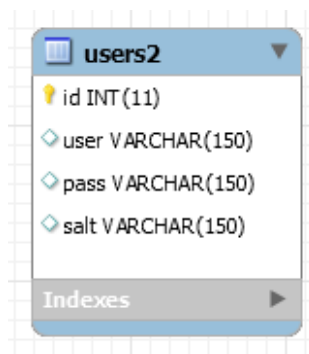
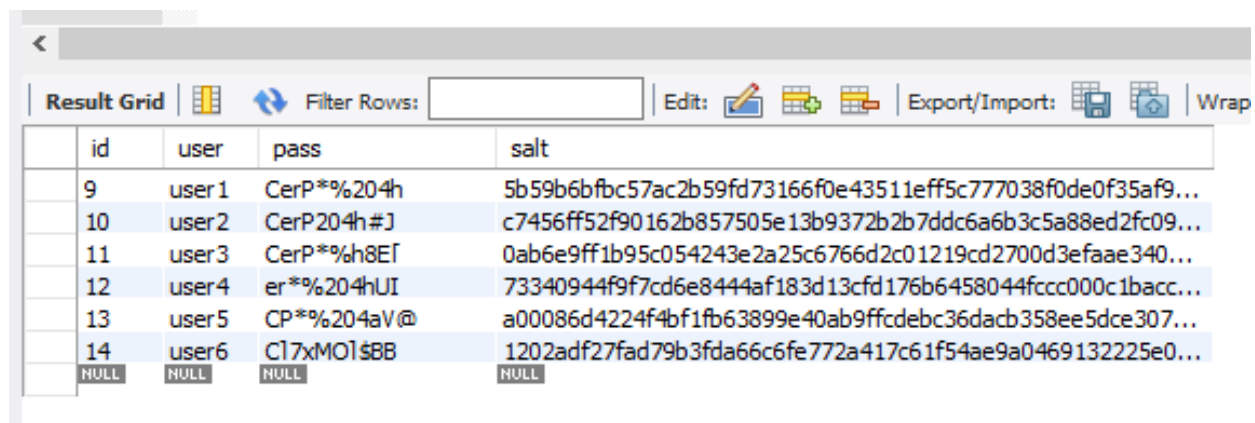


Figure 8 – Prototype's database Entity-Relationship model

Figure 8 describes the table “users2” necessary for the demonstration: “id” is the id column for the Primary Key; “user” corresponds to the username; “pass” is the password column; “salt” is the necessary salt.

For the sake of demonstrating how the prototype works, the following screenshots of the database will store a password that is not hashed, for the result to be understandable enough. In addition, the parameters of defining a strong password will be at least 1 lowercase and at least 1 uppercase letters, at least 1 number and at least 1 special character, with no less than 10 characters in length. A password to satisfy these needs is “CerP\*%204h”, which I will call given password.



id	user	pass	salt
9	user1	CerP*%204h	5b59b6bfb57ac2b59fd73166f0e43511eff5c777038f0de0f35af9...
10	user2	CerP204h#J	c7456ff52f90162b857505e13b9372b2b7ddc6a6b3c5a88ed2fc09...
11	user3	CerP*%h8E[	0ab6e9ff1b95c054243e2a25c6766d2c01219cd2700d3efaae340...
12	user4	er*%204hUI	73340944f9f7cd6e8444af183d13cfd176b6458044fcc000c1bacc...
13	user5	CP*%204aV@	a00086d4224f4bf1fb63899e40ab9ffcdebc36dacb358ee5dce307...
14	user6	C]7xMO]\$BB	1202adf27fad79b3fda66c6fe772a417c61f54ae9a0469132225e0...
	NULL	NULL	NULL

Figure 9 – Table entries

Figure 9 is a screenshot of the database taken to better demonstrate the functionality of the prototype:

- First scenario:  
input – given password, result stored – given password (CerP\*%204h);
- Second scenario:  
Input – given password without special characters, stored result – “CerP204h#J”;
- Third scenario:  
Input – given password without numbers, stored result – “CerP\*%h8E[“;
- Fourth scenario:  
Input – given password no uppercase, stored result – “er\*%204hUI”;
- Fifth scenario:  
Input – given password no lowercase, stored result – “CP\*%204aV@”;
- Sixth scenario:  
Input – letter “C”, stored result – “C]7xMO]\$BB”;

Mention 1: all instances can login with the username and the input password, demonstrating that the prototype is successful.

Mention 2: if the input field is left blank, the credential creation process will not iterate.

Mention 3: all the stored results are marked as “strong passwords” when using the measurement tools presented at 2.4. *Strong passwords, role and necessity*, proving the

prototype security considerations are not inferior to the already existing methods. To sum up, this prototype allows the usage of weak passwords and handles them in such a way that the result stored in the database is that of a strong password. When authenticating, the user will type in his easily remembered password and the system will handle it under the same conditions as in the creation step, so the user will have no problem accessing the account, as long as the passwords match.

## 4. VALIDATION OF PROTOTYPE

The prototype that I have presented along with the improvements that it brings into the field can be looked at from two perspectives: the security and the usability.

### Security

When considering the encryption aspect, the prototype's passwords that are stored in their hashed version in the database are not inferior in any way to the old fashioned method, since strong passwords are agreed upon as containing a variety of characters of a specific length, which both storing methods do that. As mentioned in the previous chapter, all the passwords comprised in the testing are marked as "strong" when subjected to the same type of measurements as the user created strong passwords.

When considering password complexity, the system's worst-case scenario was that the password input during the account creation phase would be nonexistent, in which case the algorithm would come up with a strong password; that has been dealt with by imposing a minimum number of characters. Consequently, if this minimum number of characters were 1, it would become risky for the user since the hacker could simply guess the password by typing in 1 character. Depending on the number of attempts registered by the countermeasure (explained in subchapter 2.2 *Attacks against passwords*), the attacker would have a number of chances (in that case 5) out of 94 single character tries. Since this may be considered by some a big risk, increasing the minimum number of characters can exponentially lower the probability of guessing the right password. If it is increased to 2, then the number of chances is 5 in 8742 possible solutions (94 times 93; mathematical formulas of combinations and permutations). The number of minimum characters can be increased at will.

Some might argue that the code that handles the passwords can be broken. However, if it is compiled in the same instance as the pre-existent common database insertion process, it is as safe as its predecessor.

Some might also argue that this approach will reduce the character search range for the hackers. However, if they were to get ahold of the hashes from the database, they would face a complex password which would require a long time to crack, as its predecessor. It would be highly unlikely for them to catch on what the prototype does, since no give away would get them "out of the dark"; seeing that the only requirement was a minimum number of characters, they would most likely think the opposite, that they are facing an unsecure system, which would render their brute force techniques useless. In the worst-case scenario where a hacker would actually catch on what the prototype

does, there is an immense number of ways to handle a password to make it stronger, which would, if anything, delay them even further.

In addition, since the prototype's algorithm is not limited by one solution, it can be re-written to handle the strengthening of the weak password in multiple ways. If this is the case, integrating instances of the prototype to different web platforms which result in strong passwords, but different combinations of the missing characters (apart from the user input), can lead to the obsolescence of master passwords. In practice: a user could use a weak password everywhere, but since the algorithm is doing the same thing differently, the user ends up with different passwords in databases. If the hacker breaks one of the databases, the rest of the accounts are still safe.

### Usability

One of the important accomplishments of this idea translated into a prototype is the release of the cognitive burden from a user when creating an account. Since the system does all the hard work for the user, the latter is no longer inconvenienced by having to remember tricky passwords. This allows him to feel at ease, knowing that his chosen familiar password is not weak anymore. From my point of view, this is a great example of user-friendliness expressed with the help of technology.

To back up the studies proving that password managers are not popular among users (subchapter 2.6. *Existing password solutions*), an argument against using password managers can be made. They act as intermediaries between users and their authentication objectives and while they for sure contribute to the convenience of not having to remember overly complicated passwords to feel safe, they gather all the secret codes in one place; for some people, this can create an anxiety feeling.

In addition, an intermediary means another link in the chain of users' data protection. As Loo, A. (2008) states, "The strength of a computer system's security is always measured by its weakest component", one can argue that a poorly built password manager can create more problems than it solves. In case of well-designed software, why complicate the chain with another link when it can be completely overlooked? All these aspects can be solved by using the prototype.

Looking at the prototype versus password managers sequence of actions with a GOMS mindset (Card et al., 1983), it is clear that the prototype provides a much higher efficiency, removing steps from the process and thus shortening the time needed for the user to retrieve the password and insert it into the corresponding field.

**Prototype:** user opens website -> user inserts weak, but convenient password -> user logs in;

**Password manager:** user opens password manager -> user logs in, searches for the password needed -> user opens website -> user must type in secure password (may lead to errors) -> user logs in.

Note: the actual GOMS test has not been conducted, as there are different password managers and the objective was not to single one out. However, it is clear from the



example above the efficiency of the prototype. If the test were to be conducted, it would result further proof on time saved, as the use of the password managers require overwhelmingly more keyboard and mouse usage.

As another argument in favor of using the prototype, not having to rely on too many tools to ensure the effectiveness of the objective may offer the users the satisfaction of owning their own actions, which leads to a cognitive impression of fulfillment.

## 5. DISCUSSION

The discussion points that will occur in this chapter treat both the impressions while working on this thesis, as well as what future could bring for this idea.

Throughout the thesis I have put belief into my idea, partly motivated by the fact that it is a current issue that affects a lot of people, me included. While doing the research for the theory and the prototype creation, I tried to create scenarios in which my prototype would fail and how to fix those in a secure manner, so as not to defeat the purpose of this thesis. However, given the time constraints, I feel like I was not able to create a perfect and fully operational model, but I am positive that this idea can set a solid basis for work that could greatly improve the user friendliness or password creation in the time to come. Therefore, an extensive testing session of my solution might provide the researchers with flaws. In any case though, since the prototype is just for showcase, to provide an applied perspective upon the idea, if it proves itself to be a failure, then there are different approaches that can be tried, starting from the same foundation as this thesis. As some of the studies presented in the thesis show, passwords have been viewed with great interest over the years and since the IT field is rapidly evolving, more and more people will engage with studying this concept. This, in turn, will hopefully lead to a possible breakthrough in the security of internet authentication.

When it comes to the aspect of motivating the user to use strong passwords, the idea behind the thesis, as well as the prototype, solve a great number of reasons identified by researchers as the main demotivators (“identity issues”, “social issues”, “double binds”, etc. – Sasse et al., 2001), as it would simply overlook those innate fears, fears that a system cannot possess.

Regarding future implementations, if this solution ever ends up out of the controlled environment, I think a thorough informational campaign must be carried out in order to inform users that the system is still safe, regardless of dropping the visible strong password requirements. This must be done in a careful manner that does not give the attackers anything to work with.

## 6. CONCLUSION

From the beginning of this thesis I set out to answer the question:

***How can we shift the burden of creating and thus remembering strong passwords from the users to the authentication system, without sacrificing security?***

I started off by analyzing passwords, what they are used for, how they are created, how they are stored and all the security implications of these actions. I researched different aspects about passwords and what actions/reactions they caused: how the rapid development of the IT field made them susceptible to attacks and how one could protect against these attacks. I tried to build a systematic approach on how they work and then, I moved on to the main actor in the problem formulation, the password's strength. I explained with solid examples why the systems need them, what they imply, and, from a usability point of view, why they are not the easiest solution to work with.

Then, I presented existing solutions for passwords and I introduced a basic login system prototype to show that it is possible to shift the burden of creating and remembering strong passwords from the user to the authentication system, without sacrificing the security considerations when it comes to storing passwords. I am aware that the prototype does not integrate all the features that protect against all known security flaws and they need to be integrated should the prototype be successful, however that is not the focus of the study.

As a last step of this thesis, I created a specific chapter which dealt with validating my prototype against the existing methods. Given the flaws that were presented through theory, these methods are far from perfect and there is still room for improvement, which is why the researchers will surely still continue in this direction.

With this, I conclude that the problem formulation and the associated research questions presented in chapter **1. Introduction** have been answered in a satisfying manner through the use of theoretical and practical knowledge put forth in their respective chapters.

## 7. REFERENCES

(Adams, A., & Sasse, M. 1999). Users are not the enemy. *Communications of the ACM*, 42(12), 40–46. <https://doi.org/10.1145/322796.322806>

(Alkaldi, N., & Renaud, K. 2016). Why do people adopt, or reject, smartphone password managers? In EuroUSEC '16: the 1st European Workshop on Usable Security Internet Society. <https://doi.org/10.14722/eurousec.2016.23011>

(Arstechnica 2020) Hacked French network exposed its own passwords during TV interview. Arstechnica homepage. Visited 28.08.2020 <https://arstechnica.com/information-technology/2015/04/hacked-french-network-exposed-its-own-passwords-during-tv-interview/>

- (Augusto, J. 2010). Past, present and future of ambient intelligence and smart environments. 67, 3–15. [https://doi.org/10.1007/978-3-642-11819-7\\_1](https://doi.org/10.1007/978-3-642-11819-7_1)
- (Ayyagari et al., 2019). Why Do Not We Use Password Managers? A Study on the Intention to Use Password Managers. *Contemporary Management Research*, 15(4), 227–245. <https://doi.org/10.7903/cmr.19394>
- (Card et al., 1983). *The psychology of human-computer interaction*. Erlbaum
- (CSOonline 2020). The biggest data breaches of the 21st century. CSOonline homepage. Visited 26.08.2020  
<https://www.csoonline.com/article/2130877/the-biggest-data-breaches-of-the-21st-century.html>
- (Dell’Amico et al. 2010). Password Strength: An Empirical Analysis. 1–9. <https://doi.org/10.1109/INFCOM.2010.5461951>
- (Erbschloe, M. 2020). *Social engineering: hacking systems, nations, and societies*. CRC Press.
- (Han et al., 2014). Password Cracking and Countermeasures in Computer Security: A Survey
- (Herley, C., & Van Oorschot, P. 2012). A Research Agenda Acknowledging the Persistence of Passwords. *IEEE Security & Privacy*, 10(1), 28–36. <https://doi.org/10.1109/MSP.2011.150>
- (Inetsolution 2020). Complex passwords are harder to crack, but it may not matter. Inetsolution homepage. Visited 30.08.2020  
<https://www.inetsolution.com/blog/june-2012/complex-passwords-harder-to-crack,-but-it-may-not>
- (Informationisbeautiful 2020). World’s biggest data breaches and hacks. Informationisbeautiful homepage. Visited 30.08.2020  
<https://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/>
- (Kessler, G. 2020). An overview of cryptography. Garykessler homepage. Visited 28.08.2020  
<https://www.garykessler.net/library/crypto.html>
- (Komanduri et al., 2011). Of passwords and people: measuring the effect of password-composition policies. 2595–2604. <https://doi.org/10.1145/1978942.1979321>
- (Li et al., 2014). The Emperor’s New Password Manager: Security Analysis of Web-based Password Managers. *USENIX Security Symposium*.
- (Lisa et al., 2010). AN EMPIRICAL STUDY OF USER AUTHENTICATION: THE PERCEPTIONS VERSUS PRACTICE OF STRONG PASSWORDS. *Issues in Information Systems*, 11(1), 256–265.

(Loo, A. 2008). The myths and truths of wireless security. *Communications of the ACM*, 51(2), 66–71. <https://doi.org/10.1145/1314215.1314227>

(My1login 2020). How secure is your password? My1login homepage. Visited 30.08.2020 <https://www.my1login.com/resources/password-strength-test/>

(Ntantogian et al., 2019). Evaluation of password hashing schemes in open source web platforms. *Computers & Security*, 84, 206–224. <https://doi.org/10.1016/j.cose.2019.03.011>

(Oesch, S., & Ruoti, S. 2019). That Was Then, This Is Now: A Security Evaluation of Password Generation, Storage, and Autofill in Thirteen Password Managers.

(Online Domain Tools 2020) Password checker online. Online domain homepage. Visited 30.08.2020 <http://password-checker.online-domain-tools.com/>

(Randomize 2020). How long to hack my password? Randomize homepage. Visited 30.08.2020 <https://random-ize.com/how-long-to-hack-pass/>

(Sasse et al., 2001). Transforming the “Weakest Link” — a Human/Computer Interaction Approach to Usable and Effective Security. *BT Technology Journal*, 19(3), 122–131. <https://doi.org/10.1023/A:1011902718709>

(Schneier, B. 2004). *Secrets and lies: digital security in a networked world* (Paperback ed.). Wiley.

(Shay et al., 2010). Encountering stronger password requirements: user attitudes and behaviors. 1–20. <https://doi.org/10.1145/1837110.1837113>

(Stallings, W., & Brown, L. 2018). *Computer security: principles and practice* (Fourth edition, global edition.). Pearson.

(Usesecure.io 2020). Security training topics for 2020. Usesecure homepage. Visited 26.08.2020 <https://blog.usesecure.io/12-security-awareness-topics-you-need-to-know-in-2020>

(Wash et al., 2016). Understanding Password Choices: How Frequently Entered Passwords Are Re-used across Websites. *SOUPS*.

(Wiedenbeck et al., 2005). Authentication using graphical passwords: effects of tolerance and image choice. 93, 1–12. <https://doi.org/10.1145/1073001.1073002>

(Wikipedia 2020 – Encryption). Encryption. Wikipedia homepage. Visited 27.08.2020 <https://en.wikipedia.org/wiki/Encryption>

(Wikipedia 2020 – Password). Password. Wikipedia homepage. Visited 26.08.2020 <https://en.wikipedia.org/wiki/Password>

---

(Wordfence 2020). Password authentication and password cracking. Visited 27.08.2020  
<https://www.wordfence.com/learn/how-passwords-work-and-cracking-passwords/>

(Wordpress 2020) Software tool, downloaded from the Wordpress homepage. Visited 31.08.2020  
<https://wordpress.org/download/>

(Yıldırım, M., & Mackie, I. 2019). Encouraging users to improve password security and memorability. *International Journal of Information Security*, 18(6), 741-759. <https://doi.org/10.1007/s10207-019-00429-y>