Password Universe Visualisation

Department of Computing

Georgi Damyanov

Abstract

This report documents the creation of my password universe visualisation (PU) which is modelled on a web application. This PU is used to visualise a given password database as the "universe" and the entries as "stars". The user is treated as the centre of this universe by choosing a keyword to compare with the database. Database doesn't necessarily need to be with passwords, it can be any combinations of strings such as an English dictionary. It's mostly aimed at password researchers and cyber security experts allowing them to see patterns thus helping them to gain further understanding of how passwords are related to one another. It can allow users to compare the same password with different databases and gain results of how users come up with their passwords. In this report we learn what the real issue is with users and their passwords, how does it affect us and how a lot of people are deceived and uninformed that their passwords are not secure. This PU proposes a solution into gaining more data in the field of passwords that can be used in order to improve password policies. Afterwards there is a description on how the code works with code examples, what decisions I made to create it and what is the potential future work.

Contents

1. Introduction

There are billions of accounts hacked every year, unhashed data being spread around the internet that can be gathered if you know where to look. For example, recently there was a leak of millions of emails and passwords stolen and a lot have been unhashed and are spreading around [1], making hacking users easier using standard password cracking techniques such as dictionary attack, brute force attack and rainbow table attack. However, there is also a problem that with such huge database of real unhashed user passwords, that many other can be potentially at risk because they have had a similar or the same password as the one that got hacked causing a leak of for example 10 million, possibly affect a much larger population. Alternatively, even if the user is aware that his password has been hacked, he will use a simple pattern to change it. This project aims at using any password database to visualise potential patterns people have whilst creating their passwords which can be used to potentially strengthen the requirements for passwords on websites. For example, research such as this can be used to improve something most websites have, proactive password checkers (PPC) about which we will talk more in Section 2. How does one decide what rules to add to their requirements for these password meters? Different password meters perform differently [2] [3]. There was a research made in 2010 stating that our current password creation policies are still vulnerable to online attacks [4] and this issue still persists as more and more users are keeping the same passwords [3], using the same patterns of creation without much knowledge of them being hacked and vulnerable, and as I mentioned even if they know, most will result to a simple pattern fix. When starting this project, I had multiple scenarios presented to me for data visualisation that I can approach such as;

1. Heat maps – generalised heat map giving the frequency of each character at each position and each password is a segmented line on the heat map.

2. Human related passwords – passwords related to geo-locations, related to a person's name, animals, plants and even religion, languages and wars.

These two are just some of the many ideas I was given by my supervisor but I chose to go along with the "Passwords in a universe" where each star is a password and the user keyword is in the middle of this universe.

The reason behind me choosing this idea is because personally I could already see the possibilities of how much this can benefit professionals, I could visualise what I want to build and how I would like to make it work, the issue was how I would achieve that. Nonetheless I believe I made the correct choice and I hope this tool can help improve and break patterns that current users have with regards to their passwords.

I believe that with the power of visualisation, it can be much easier to actually see patterns and collect data otherwise left unseen. The use of visualisation techniques have been used for a very long time from maps to visualise one's location to graphs, to visualise one's data. In a short Q&A video, Simon Samuel, Head of Customer Value Modelling for a large bank in the UK, he answers the question "How important is data visualisation" by saying:

"Visualisation is fundamental and it will be increased with its usage going forward. The executive of the bank where I work currently demands more visualisation tools to help them support their insight and analysis and also to accelerate their understanding".

Visualisation can identify areas needing for additional attention much easier and simpler for the human brain to see and identify. Any visualisation tools that can help understand data and make important decisions on how to detect patterns and do our best to correct them can be useful [1].

In this paper I will be talking about:

1. The issue, how users don't understand the risks of using same passwords everywhere, how even if some of their passwords are slightly different, they use simple well known patterns to differentiate them which can easily be spotted using my visualiser.

2. My password universe, what can my application do, how can it potentially help and show some examples of known patterns, how does using edit distance improve on today's password meters algorithms which look at specific characters such as "symbols, capitals, variety of letters" etc. and treats them with a high difference.

3. Results, I ran some benchmarks to look at speeds varying on different database sizes. How the size of the keyword affects the speed.

4. Issues, what are some of the currently known issues and how they can be improved.

5. Conclusion, what I learned in the end, how helpful this can be.

2. Related Work

Simple text passwords are the most used form of identification and grant permissions. Even though there are many other ways of identifying yourself such as face recognitions, thumb prints, key cards etc. most places still use the standard password. By default if something is used the most, it should be the most secure in terms of identification but in regards to password cracking, its currently a big issue because of how users create their passwords and once leaked somehow, even if warned, they persist to make simple changes instead of making a serious change in their passwords. There are password checkers to try and make users to think of newer passwords and it has worked in the past but the issue persists that users keep creating similar passwords.

Most users when creating a password have fallen into some known patterns that they use assuring themselves that their passwords are secure. An issue comes from the overly rated password meters which allure users to believe that since it satisfies the meter, their password is surely secure and hard to crack [2] [3].

For all examples below in this section, I will be using a standard initial password: "password".

There are a few common patterns that people use to make their password more secure. For example, capitalising the first or last letters of their password. One of the most common password meter requirements is to have one upper and one lowercase letter. Users are so used to their current passwords that they choose to adapt their current password by just capitalising a letter in their password that would be easy for them to remember but yet satisfy the password meter. For example, turning "password" into "Password", "passworD" or "PassworD".

Furthermore, there is also the "1" or any short combination of following numbers at the end of the password. Again this comes from the standard requirement of password meters requiring at least one number into your passwords giving false sense of security to users thinking their password is securely protected.

For example, changing the initial password to "password1", "password12" or "password123321"

Finally, there is another common pattern people use as an alternate or with the one stated above which is replacing letters with similar looking numbers such as (t → 7, s→ 5, I → 1, o→ 0). For example, "password" would turn into "pa55w0rd" or "pas5w0rd".

Combining all of these tricks the standard "password" would turn into potentially something like "Pas5w0rd123" which from the normal users perspective makes him believe that this change of his password. Password meters are evolving but they are doing small changes which just pushes users into more patterns which eventually makes no difference of their password being cracked.

These are some of the most known patterns and if one of a user's old passwords has ever been compromised by a leak or any other way, there is a high chance the person trying to crack you will try to follow all of these pattern combinations. These examples show how users are overly relying on just satisfying a password requirement leading to ideal security for their passwords. This calls for a drastic improvement on password meters, something has to change and the ideal way is to notice as many patterns as we can that users are creating and proposing solutions to these patterns so we can tighten securities.

2.1 Password Security

When it comes to password security a big issue is that the majority of users don't actually study about password patterns or any password safety, they highly rely on the providers (where they register at) to deal with all the security and they commonly create passwords that are easy for them to remember not thinking that the issue is that many other people will think of the same thing. There has been research on semantic patterns showing that like to use dates or certain words for their password that they find close to them and easy to remember. The research is based on the Rockyou table and they found out that about 4% of the passwords are date related only and can be cracked easily with a dictionary attack with about 200,000 entries in a table [4]. This proves to be clear issue and requires the improvement of password policies and give appropriate guidelines for users to improve their passwords. Through the years one common method of trying to tackle this issue is using a proactive password checker (PPC) which is the tool websites commonly used to measure user password strengths and make sure the user password retains certain conditions and those conditions are based on the password meter function (PMF) embedded in the PPC.

People have tried improving PMF's which are generally based on certain password composition policies which restricts the usage of simple weak passwords that are highly vulnerable to statistical attacks and add more criteria to them or changing the algorithms to portray how strong the password actually is [5] but even with all these changes, users just start creating more patterns that do not get addressed which causes PPC to display password as strong giving a false sense of security to the user.

For example:

Let's take a short combination of my date of birth (DoB) and name and satisfy a PPC, something that is close to me and easy to remember. I will be using "9970707gG!1" Where "997" comes from the year I was born 1997, the "0707" are day and month, "gG" comes from my first name letter, and just capitalising it to make the password requirements happy, and "1!" which is just the first number on my keyboard to again just satisfy the password checker. Now this might seem secure but in reality, this is just a mix of simple patterns to update my original password "9970707" which is what I had at the beginning and it's fairly easy to guess if the attacker has any knowledge of me but the PPC won't know that and it would say it's a good password, even without any knowledge, there is a high chance such password is on some rainbow table which are simply put tables with common leaked passwords.

PPC's do work and still work in general about most things but something has to be done and improved in the field for users to create more secure passwords. PPC's mostly act as a password guideline at the moment.

Alternatives are password managers (PM) those are tools that allow users to store all their password behind a single master password and most will auto-fill the passwords for the users on the web. The tools will generate a good password for the user and store it in a hashed format. PM's do partially work but not many people know about them and if they do, they still don't bother using them. Good examples of PM's right now are 1Password and LastPass which are very reliable and simple to use and setup for all desktops and mobile devices.

2.2 Password Visualisers

There are other examples of password visualisation where researchers have tried to showcase password evolution again by using Levenshtein distance by using the top leaked passwords of websites. Their results show that passwords also like humans have aspects of communities and social networks by being linked in some way to one another. It helps them visualise chains forming from different password distances and the evolution of one password to another. Using these visualisation techniques they have tried to also determine how users create their passwords and analyse their password patterns to improve current password policies and give appropriate guidelines for the aiding of more secure passwords [6]. Alternative visualisation technique is also heatmaps which were used in a paper to visualise user PIN's by plotting on the X-axis the first two digits of their pin and on the Y-axis the last two digits of the PIN [7]. Those heatmaps allowed to easily view patterns on which two digits are used the most at start and end by forming lines on each axis. An example for spotting patterns on heatmaps would be where the line is much sharper it would represent a frequent usage, if two lines are crossing each other, the cross-section of those lines would be a commonly used full pin number.

3. Design

Unlike password checkers which use some algorithm to detect the total amount of symbols, numbers or other variables within the password to determine its strength, this PU uses an edit distance using Levenshtein distance formula to determine the total number of mutations a password needs to take to go from A to B or in this case, from the keyword a user typed to every database entry. All databases that I have used are online and are unhashed plain text leaked passwords that were made available for research purposes.

3.1 Levenshtein Distance

How Levenshtein distance works is it looks at the first string, in our case it's the keyword a user has entered. Let's say its "password". Then it looks at the string inside our database, "p@sword". As visualisation look at Figure 1 below, the algorithm would look at each character in in both strings and one by one determine if the character matches, if it has to be changed to match or a new character has to be added instead. In our case, the edit distance in this example would be 2 because to get from the initial password, to the second password, you need to:

1) Change the 'a' character to '@'.
2) Remove the second 's' character.

p	a	s	s	w	o	r	d
p	@	s		w	o	r	d

Figure 1: Displaying how Levenshtein distance works

The benefits of using Levenshtein distance is that it completely bypasses all the tricks users use as patterns and it's much easier to see what they are doing. How it works is that it looks at all changes that need be done to get from one password to another so if your password is "password" and you change to for example "p@ssword" or "p4ssword" since both "@" and "4" are known replacements for "a", a PPC it would determine them in different strength since it would believe that having "@" instead of "4" or a would be better, alternatively if you already have a symbol but no number, it would think having "4" better than "@" but in reality, it's a single character change to your password.

In the password universe, Levenshtein distance is used for the radius for each password, and the angle is found by the alphabetical sorting. I also add different colours to each radius based on its distance as shown by Figure 2 to try and get an easier understanding from the visualisation. When I test "9970707gG!1", the

password we used in earlier examples Figure 3, it can clearly be seen that the password in reality, is very similar to other passwords. A lot of the colour is red, distance is between 5 and 10 inside that section. The purpose of this is to show that there are not so many changes to do from a password to reach to another user's password. Even though a user's password might not be in danger. Other passwords which were leaked can be in close range to the password entered.

```
switch (true) {
    case (radius < 5) :
        hexString = "#8000" + (radius*5);
        break;
    case (radius < 10):
        hexString = "#ff00" + stripSingleDigit;
        break;
    case (radius < 15):
        hexString = "#0066" + stripSingleDigit;
        break;
    case (radius < 25):
        hexString = "#0099" + stripSingleDigit;
        break;
    case (radius < 35):
        hexString = "#33cc" + stripSingleDigit;
        break;
    case (radius < 50):
        hexString = "#00cc" + stripSingleDigit;
        break;
    default:
        hexString = "#00ff00";
}
return hexString;
```

Figure 2: Switch code for colour fills on circles

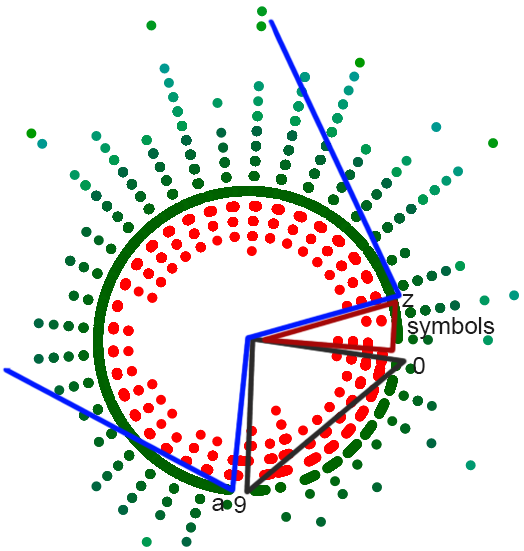
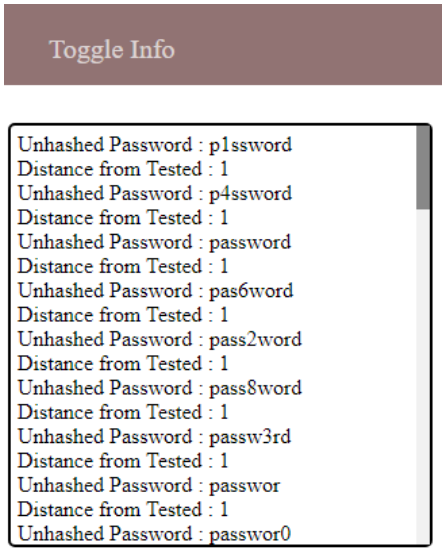


Figure 3: Diagram of "9970707gG!1" password

This diagram can show us that there is a clear pattern with similar passwords which were hacked. I have added the blue, red and black lines representing the alphabetical ordering from 0-9, a-z and symbols sections for better understanding.

3.2 Information Table



The diagram PU creates allows the user to click on an area and see what the passwords are in a radius. There are no passwords directly under each other, but there are a lot of passwords in a very small area with minimal difference between each other for the X and Y co-ordinates, so if a user wants to see what's in a certain area of dots, he can click on them, and on the side there will be an information bar that can be toggled on or off. If you look at Figure 4 you can see a result of a radius when the keyword "password" is entered.

Figure 4: Information box of passwords

This information shows leaks of passwords all spelling "password" in a different variation by using some of the patterns we mentioned above in Section 2. Even without using the information box we can determine by just the visualisation that there is some pattern in an area, comparing it to the rest of the visualisation it stands out and the human brain is highly likely to notice that much easier in this visualisation than just looking at the passwords as a plaintext.

The way this function works is it allows for a user to click anywhere on the screen, if he clicks on a dot, it calls a function which creates an array of all dots in the area of size determined inside the code and appends the information I have decided to display inside the information box table which currently is “Distance” and “Unhashed Password”, alternatively if there were results previously already displayed, they will be removed and then the new ones will be appended.

To aid for the understanding of what data is actually being portrayed, I have made it so that the area of dots that being selected are all coloured in dark blue as shown in Figure 5.

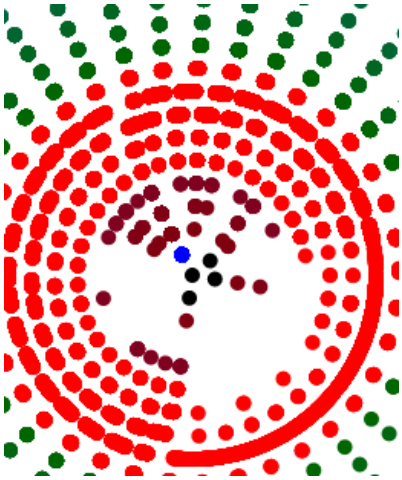


Figure 5: Highlighted section for the keyword “password” displayed on Figure 4

3.3 Sorting algorithm

Currently the sorting algorithm is alphabetically. The way the graph gets plotted is by me calculating the edit distance explained in Section 3.1, then I decide how many sectors my universe will have. I have a-z, 0-9 and symbols will be treated as 1. This makes total of 37 sectors. I create an array of 37 arrays. Each array inside my main array represent a sector of a circle. Then I iterate through the entire database and take the 1st character of each entry in my database, look it up on the ascii table and determine in which array I will push it into based on that. Once I have determined the array, I will put it inside. Once all entries in the database are placed inside the appropriate sectors, I sort each sector alphabetically using the JavaScript sort function. Now I have all my sectors with all my passwords sorted alphabetically, I now need to get their X and Y co-ordinates. I iterate through my array of sectors and each entry in every sector, I then calculate the X and Y by first finding out what the angle of the password will be. Once I have my radius and my angle, I can use Sin and Cos rules to determine what the X and Y co-ordinates will be. Once you get the X and Y co-ordinates I use the d3js JavaScript framework to plot the data on the webpage inside an SVG container (the password universe) and creating each data point as a circle object (the stars in my password universe). This data can then be used in away way desired but can also be downloaded since I have provided a download feature which would take the entire SVG data and save it on your computer in an SVG format which it can then be used to store the image and represent it using another application or collect different specific data points that could be used for analytical reasons.

```
for(var i = 0; i<sectors.length; i++) {
    // Look at all the values for this sector
    for(var j = 0; j<sectors[i].length; j++) {
        var radius = (sectors[i][j][1])*15;
        // Now determine the angle by looking using PI formula but also including the total
        // sectors to make sure they don't overlap
        var angle = (Math.PI * 2) * (i/maxSectors) + ((Math.PI * 2) / maxSectors / sectors[i].length * j);
        var getY = (Math.sin(angle)*radius);
        var getX = (Math.cos(angle)*radius);
```

Figure 6: Code for the co-ordinates generation

The reason for the *15 in the radius is a naïve attempt to scale the initial displayed graph. This value is not used anywhere else and does not affect the correctness of position on the graph but there are potential issues with doing it this way in which I will talk in a later section about all the issues in the code and how can they be fixed.

4. Results

I will use this section to showcase data with different passwords to show clear difference of weak versus strong, show potential patterns and what kind of details can researchers analyse. I will be using three different databases. The 1st database I will be using is from Hotmail and it contains 8930 passwords, 2nd database is from MySpace and it contains 37144 passwords and finally I'm using the phpbb password which has nearly 185,000 passwords. I need to note, all these passwords are available to use online for educational purposes. These passwords were leaked or stolen from sites.

I will write a keyword on each database to see if we can see any patterns between these databases.

Example 1:

"123456789" a very simple pattern of sequential numbers.

a) phpbb b) hotmail

Figure 7.1: Patterns of two databases with the keyword "123456789"

If we can shortly refer back to Figure 3 to check how the alphabetical pattern looks like, we can notice that there is a clear line forming around A-Z potentially hitting some of the symbols suggesting that potentially there are a lot of passwords which start with letters but end in numbers. There seems to be another pattern in both images as well, you can see that within the numbers location, it nearly forms a triangle because of the small amount of empty space. This pattern is a lot more noticeable in the hotmail database, if you look closer you can see that the colours are purple, this is due to the colour settings seen in Figure 2 suggesting it's around the 0-5 area. If both pattern are downloaded and overlapped you can get a result showcasing how both databases with completely different sizes can form similar patterns showing how users follow their own set of composition policies.

Using this visualisation we can clearly see that there are patterns formed but if we want to analyse it any further, one can start using the Information table I have added to start looking at what these passwords actually are, how are they created and why do they appear where they are which can reveal to common password patterns and help improve PMF's to prevent from users creating such passwords or at least try and give additional guidance as to why such password patterns are not ideal.

Example 2

“password” a very common password usually used as a placeholder password

a) phpbb b) hotmail

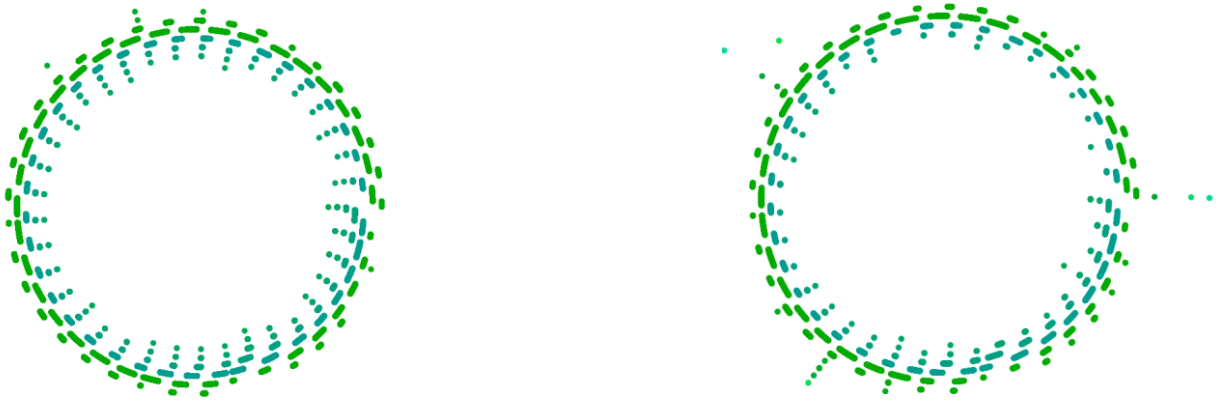
Figure 7.2 Patterns of the databases with the keyword “password”

With this example I am trying to showcase the exact opposite from typing a statistically known sequence of numbers, to a statically known sequence of characters which represents the exact opposite pattern of how users create a lot more similar passwords starting with numbers and potentially finishing with letters.

Alternative pattern that is visible a lot more on phpbb than hotmail since the database is much larger is that around the middle, there is an extreme amount of similar passwords that have been cracked, its nearly covering the entire area which means that a lot more users have had this password or very similar ones leaked than the previous example.

Example 3

“7E0@67au1h21G*vR$Bdp0” is a password generated from a password manager “LastPass”



a) phpbb b) hotmail

Figure 7.3 Patterns of the databases with the keyword “7E0@67au1h21G*vR$Bdp0”

This example is a clear showcase of a much more secure password, the patterns look absolutely different, you can easily spot that nothing is nearby, the middle is an empty void and showcases a password entropy, and all the passwords are far away forming a nicely patterned circle.

There is a huge disadvantage of using a password as this, and that is that no normal person would easily remember this, this password has to be written somewhere and that's the advantages of owning a password manager.

All of these examples show that different passwords, can create different patterns and these patterns can showcase variety of things such as how secure a password is by looking at any nearby stars in the PU of each keyword or alternatively common password patterns user create for themselves by trying to create an easily memorable password for much easier access. No matter what the pattern showcases, it can be used to create better password policies for PPC's or alternatively they could be used for better feedback support to the user if they use some sort of a PPC in the future which can tell them stuff such as "we have detected a common policy users used", currently we don't have enough data to provide such suggestions as definitive but this application can help contribute to analysing multiple passwords and collecting such data.

5. Known issues

Whilst reading this paper there is a possibilities that issues might have been noticed and there are a few issues that I would like to address and how they could be fixed.

5.1 Password case

Starting with a very important issue is that all passwords are actually being changed to full lowercase. The reason this was done at start is so that it's easier to put them into categories, where I mention on section 3.3 where I create the sectors, for the ascii table to make it simpler I made all characters into lowercase, this can easily be fixed if the line where the characters are being turned to lowercase is removed and adapt the total size of the total sectors from 37 to 63 so we can add space for the 26 uppercase characters. We will also need to change how lowercase and uppercase are now ordered, we will want aA bB cC etc. so we will need to play around a bit with the maths to order them. Alternatively, it will be much easier to have for example A-Z a-z or vice versa since it will be a bit simpler to implement.

5.2 Initial scaling

As mentioned and showed at Figure 6, the initial scaling at the moment is *15 and it works in my examples but if someone decides to test a database resulting to very large edit distance results, it will make it absolutely huge and hard to read because the patterns will be very distanced out. I had a solution on this to get the smallest radius distance or the largest radius distance in the array and then decide what you would want the min-max to be and set all passwords by the amount you set the first password chosen. So for example if the smallest password is of radius 5, and you want the minimum to be 15, add 10 to the password and to all other passwords to scale them to reach a minimum, you can do the same with maximum, to get the largest value and scale it down to the largest radius decided on.

Alternatively, d3js provides a scaling algorithm that can be used instead by giving an array of values that you want it to be scaled at and use those values for scaling.

5.3 Centred view

Currently, the way the view gets centred is by getting the largest and smallest X and Y co-ordinates and altering the padding using JavaScript CSS. I couldn't figure out a way how to keep the circles inside a div box any other way so they kept escaping the container I have added them into and when they keep resizing any other way I tried doing this has failed.

An example of where this is an issue is if Section 4.2 is actually ran, you will notice how the scroll bars are all over the place and its quite hard to locate where the graph actually is because there are massive outliers highly skewing the view on the page.

If the resizing is to be kept as it currently is, another way to solve this issue is create a function to remove massive outliers by giving some sort of way a maximum size of the passwords to be used and removing all other passwords.

5.4 Call stack

So the issue here seems to be that when I load a huge chunk of database, my code exceeds the stack because of the nearest.js code. Unfortunately I can't find exactly where the bug is, I got partially the code from a jsfiddle (credited in the code) I found, then looked at what the code does and changed it so it fits my scenario. The original code would just highlight area in a radius around the mouse, I had to make it instead to on click get the dots in a radius, pull their data and place them inside the box.

I cannot think of a fix for this issue unfortunately, the only think I can think of is to re-write the function or think of a different way to potentially pull the data. A user is still able to download all the data using the download function if he requires and the visualisation still renders, it just cannot get all the individual points.

6. Conclusion

In this section I would to talk about how this project can be taken further, what some of the limitations were and my personal reflection on how I found the project, how it affected me and what I learned.

6.1 Future Work

I took this project and created the basis of other projects which can be potentially way more useful and with time, this project can become an amazing tool but there are certain things that have to be made.

6.1.1 General Improvements

I want to go over a few general improvements that can definitely improve the current code as it is, make it easier to use and more user friendly. After those improvements, I will be talking about how the code can be adapted to visualise more patterns.

6.1.1.1 Zoom feature

The zoom feature at the moment its very basic, all it does is that it scales all X and Y co-ordinates by 1.5 depending if you click zoom in, it multiplies and if you zoom out, it divides. What I would envision ideally be for the user to be able to select an area with the mouse, take that area and zoom only that area by keeping the rest normal zoom. Maybe it could create a new div at the same location the area is but make that div cover the original by allowing the user to zoom in or out depending on how they wish it to be.

Another alternative of the zoom is to create a dropdown scrollbar, that scrollbar can be scrolled up or down and create an ajax zooming potentially where it doesn't require the user to wait for all points to be re-rendered with the new parameters. To achieve this I believe the d3js scaling could work, but I have not tested it.

6.1.1.2 Selecting data points

I showed how one can use the information box to gain data about the databases and try and reveal potential patterns in Section 4 but the issue is that you can only select a certain area as briefly explained in Section 5.4 on how it works, I think it would be better if a user can potentially either:

1. Select multiple dots, as many as he wants and then click a button to gather all the data from the dots selected and display them, this way a user can select multiple dots from different areas and contrast pattern information much easier.
2. The same alternative as the zoom feature in 6.1.1 where a user can select an area of different dots and once released and selected the area, all dots inside will display their data.

6.1.2 Adding Features

Now I want to talk about future works with new features that can improve the overall usage of the application by the user. These are features that can potentially improve user experience or completely change the patterns a user is looking at with the databases.

6.1.2.1 Uploading databases online

If this application is to be made public for users to use all over the world, we would need to allow the users to upload their own databases using the online application instead of having to download the code onto their computers. Furthermore, possibly allow different patterns of databases to be uploaded, currently the database is a .js file with a function name and parsed through the html for the d3js file can read the data using basic JavaScript and users might not understand this, so allowing .json , .sql or other file extensions.

6.1.2.2 Multiple results and Overlap

Ideally a user would be able to choose multiple words, possibly from multiple databases allowing to store multiple results on the same session and overlapping them. If they can overlap, they will also need the ability to change the colours manually from the online application on each graph, furthermore they should be able to for example keep or remove the entries of both databases from the results that exactly overlap or are within a very close proximity from one another. This way you can for example see the same patterns appearing in different databases or look at the variety of passwords from both databases that can result to such edit distance and allow for a bigger pool of data to be analysed and help determine patterns people use.

6.2 Personal Reflection

Overall throughout this large individual project, I have learned a variety of things, face a lot of challenges and I would like to write a short description on what I have learned.

To begin with, I've never done any visualisation before, I did some JavaScript at my placement, not much but I was interested to give it a go since the framework for visualisation was in JavaScript, which was one of the reason I picked this project. I found out the visualisation framework Is very powerful and what I did with it is just the surface. I used d3js but there are other alternatives like p5js but the reason I chose d3 is because its newer, I feel like gaining experience in this framework could benefit me more in the future and I had a lot of fun learning what I can do with it from the first prototype I created that replicated the visualisation design of my supervisor's paper [8] to my final application result which amazed me how accurately data can be represented and how much easier it makes it to read and spot patterns.

Security wise, personally I never really thought too much of my password security, I had a password which I was fairly sure it was sure-proof but learning about all these patterns people make, testing all kinds of passwords on this visualiser both for personal reasons and research reasons, it strongly made me realise that I was just another person following the same simple patterns. I now personally did some research on password managers, picked up one that I think fits me and use their password generation for my password, as a great example of a potential diagram of such password you can recall Figure 5.1.

7. Challenges

Biggest challenge I found with this assignment is understanding how I will be plotting the dots. Figuring out the algorithm as it is at the moment took a long process. I tried applying Dimensionality Reduction algorithm to distribute the X and Y co-ordinates but I couldn't understand how to apply it properly. After a while of trying to apply this, I and my supervisor decided to leave it and instead try to use a different method to find the X and Y co-ordinates and plot them. I instead try and find radians so I can use the radius and radians to convert into X and Y co-ordinates using JavaScript functions. Once I got the X and Y co-ordinates, I still had to somehow make it so that they can actually create patterns and not go right one under the other and this is where I create the sectors so that each character goes into its own sector, and

when I create the radius which I use for X and Y co-ordinates I divide it from them so that they can always be changed slightly based on their position in the array of the sector.

The second challenge is finding out how to change the colour for each dot. Ideally it would be so that it can somehow change its colour on its own without having a limit but instead I had to do it with a switch statement, its slightly hardcoded but at the same time its slightly adapting its colours if you can see Figure 2 I am changing it by a single value that I get from the radius.

8. Acknowledgements

I would like to thank my supervisor Dr Shujun Li for supporting me throughout this project by giving variety of ideas and compromising on certain aspects of the code due to my unexperienced with certain technologies.

Works Cited

[1] M. Daradkeh, C. Churcher and A. McKinnon, “Supporting informed decision-making under uncertainty and risk through interactive visualisation,” 1 February 2013. [Online]. Available: https://dl-acm-org.chain.kent.ac.uk/citation.cfm?id=2525496. [Accessed 20 March 2019].

[2] T. Emin, “Cracking More Password Hashes With Patterns,” August 2015. [Online]. Available: https://www.researchgate.net/publication/276113338_Cracking_More_Password_Hashes_With_Patterns. [Accessed 20 March 2019].

[3] “What 10 million passwords reveal about the people who choose them,” [Online]. Available: https://wpengine.com/unmasked/. [Accessed 20 March 2019].

[4] R. Veras, J. Thorpe and C. Collins, “Visualising semantics in passwords: the role of dates,” 15 October 2012. [Online]. Available: https://dl-acm-org.chain.kent.ac.uk/citation.cfm?id=2379702. [Accessed 27 March 2019].

[5] H. Tupsamudre, V. Banahatti and S. Lodha, “POSTER: Improved Markov Strength Meters for Passwords,” 24 October 2016. [Online]. Available: https://dl-acm-org.chain.kent.ac.uk/citation.cfm?id=2989058. [Accessed 20 March 2019].

[6] G. Xiujia, C. Haibo, L. Xuqin, X. Xiangyu and C. Zhong, “The Scale-free Network of Passwords : Visualisation and Estimation of Empirical Passwords,” 25 November 2015. [Online]. Available: https://arxiv.org/abs/1511.08324. [Accessed 27 March 2019].

[7] D. Wang, Q. Gu, X. Huang and P. Wang, “Understanding Human-Chosen PINs: Characteristics, Distribution and Security,” 2 April 2017. [Online]. Available: https://dl-acm-org.chain.kent.ac.uk/citation.cfm?id=3053031. [Accessed 28 March 2019].

[8] S. Li, N. Aljaffan and H. Yuan, “PSV (Password Security Visualiser): From Password Checking to User Education,” 13 May 2017. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-319-58460-7_13. [Accessed 23 March 2019].

[9] T. Hunt, “The 773 Million Record "Collection #1" Data Breach,” 17 January 2019. [Online]. Available: https://www.troyhunt.com/the-773-million-record-collection-1-data-reach/.

[10] M. Weir, S. Aggarwal, M. Collins and H. Stern, “Testing metrics for password creation policies by attacking large sets of revealed passwords,” 4 October 2010. [Online]. Available: https://dl-acm-org.chain.kent.ac.uk/citation.cfm?id=1866327. [Accessed 17 March 2019].

[11] S. Egelman, A. Sotirakopoulos, I. Muslukhov, K. Beznosov and C. Herley, “Does My Password Go up to Eleven?,” 2 May 2013. [Online]. Available: http://delivery.acm.org.chain.kent.ac.uk/10.1145/2490000/2481329/p2379-egelman.pdf?ip=129.12.11.80&id=2481329&acc=ACTIVE%20SERVICE&key=BF07A2EE685417C5%2E465674151BD0BA75%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1552856965_fd3233b0869e0192d20f302f14c. [Accessed 17 March 2019].

[12] B. Ur, P. G. Kelley, S. Komanduri, J. Lee, M. Maass, M. L. Mazurek, T. Passaro, R. Shay, T. Vidas, L. Bauer, N. Christin and L. Faith Cranor, “How Does Your Password Measure Up?,” 2012. [Online]. Available: https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final209.pdf. [Accessed 10 March 2019].