**Abstract - The aim of this project is to create a password universe (PU) modelled on a web application. This PU is used to visualise a given password database as the “universe” and the entries of it 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.**

## **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 visualize 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, password meters. How does one decide what rules to add to their password 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 [5].

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 visualizer.

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. User patterns, no true password entropy**

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.

### 2.1 User Patterns

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 [6] [7].

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.

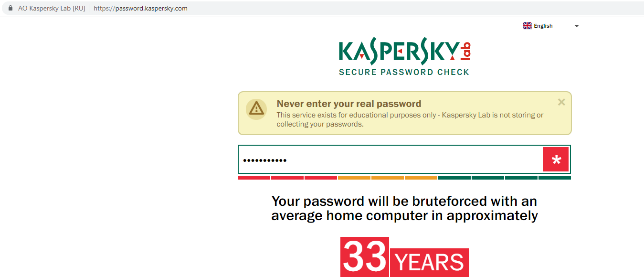
### 2.2 Adapting password meters

People have tried improving password meters and add more criteria to them or changing the algorithms to portray how strong the password actually is [8] but even with all these changes, users just start creating more patterns which are not addressed which causes users to see what the strength meter says on the password meters thinking its highly secure password but in reality, its a small change.

For example:

Let’s take a short combination of my DoB and name and satisfy a password meter. 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 person trying to crack me has specifically targeted me and can figure out my DoB quite easily from social media.

Let’s see what some password strength checkers say about this password change.

1. [https://password.kaspersky.com/](https://password.kaspersky.com/)



2) <https://howsecureismypassword.net/>





These password strength meters were taken from the top results Google offers when you search for “strength password checker” and which is what most users would probably end up using if they want to check the strength of their password. And as complex as this example password might look “9970707gG!1”, if my old password “9970707” was leaked, someone who has knowledge of these simple patterns can easily crack this password and gain access showing how misleading password meters can be.

## **3. The Password Universe**

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, I use 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:

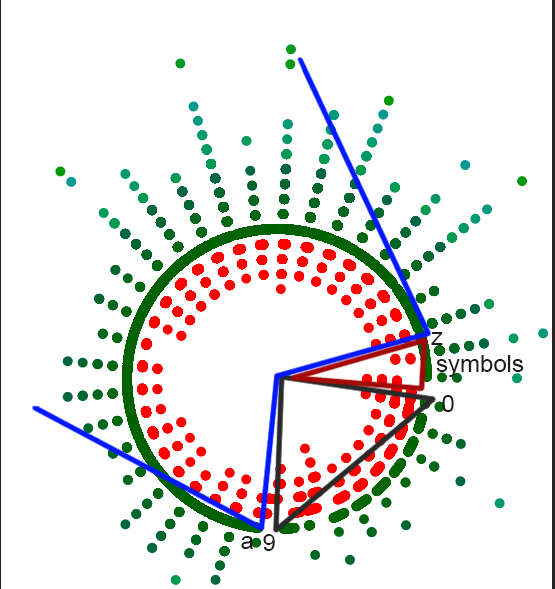
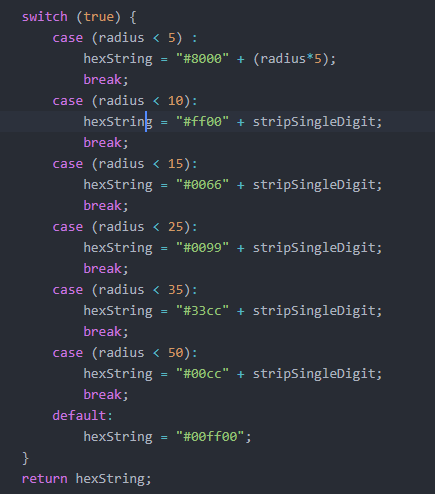
1. Change the ‘a’ character to ‘@’.
2. Remove the second ‘s’ character.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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](mailto:p@ssword)” or “p4ssword” since both “@” and “4” are known replacements for “a”, a password meter 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 colors 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.



*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.

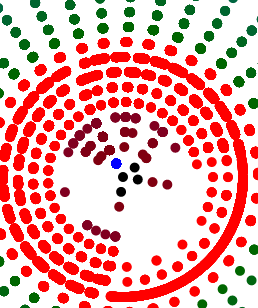
### 3.2 Reading the diagram

The diagram allows the user to click on an area and see what passwords are in the radius. There are no passwords directly under each other, but because there are so many passwords, in the current database I am using, there are nearly 215,000 passwords with removed duplicates, there are a lot of passwords in a very small area with minimal distance difference between each other for the X and Y co-ordinates, so if a user wants to see whats 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 with data for the passwords. 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 clearly shows hacked passwords all spelling “password” in a different variation by using some of the patterns we mentioned above in Section 2.1. If we focus now back to Figure 3, we can see that between a-z there seems to be some pattern of passwords that forms an entire green line. This is clearly some pattern that users have created and that we can analyse slowly by using the information box. Even without using the information box we can determine by just the visualization that there is some pattern in that area, comparing it to the rest of the visualisation it just 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.

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*

All the examples I have shown right now are in the form of weak and predictable passwords so I would like to display what happens when I use a password generator set up to create a password with length of 16 including symbols, numbers, uppercase and lowercase characters which came up with “zxS26&UmDO3@GuOH”



*Figure 5.1: Result of a much more secure password*

This is an example you would ideally want to get as a user and researchers can use the patterns on the side to notice what makes this password so unique compared to passwords who have already been cracked. Alternatively they could determine patterns such as, commonly used first letters, commonly used words etc. There are not many possible patterns researchers can detect using my naïve way of sorting in which we are going to talk about next.

### 3.3 Sorting algorithm alphabetically 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.

### 

*Figure 6: Code for the co-ordinates generation*

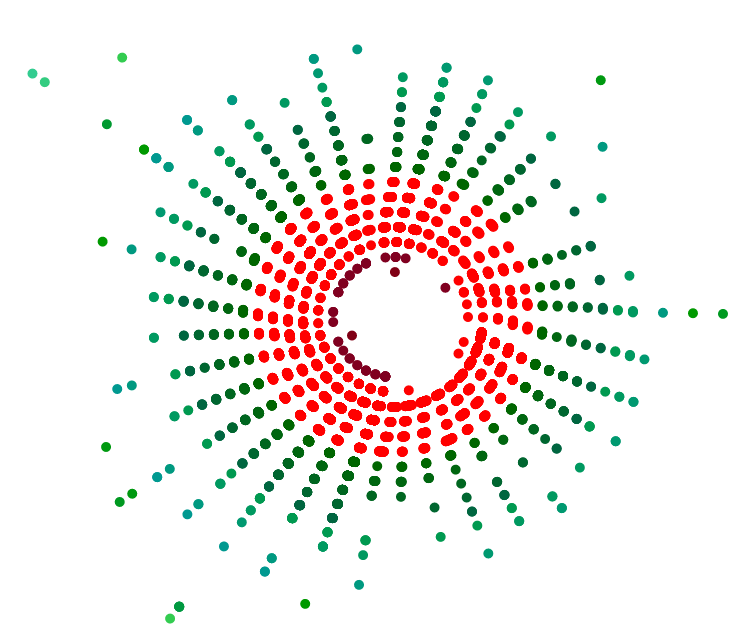
The reason for the \*15 in the radius is a naïve attempt to scale the initial displayed graph. This value is not used anywhere else and does not affect the correctness of position on the graph but there are potential issues with doing it this way in which I will talk in a later section about all the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

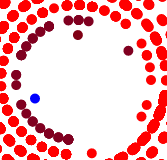
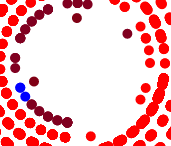
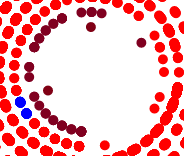
I will want to show potential patterns of names. I will be using my name “Georgi” as example of writing in the password field and I want to see what kind of potential patterns I will see. To note, I know that “Georgi” is not a secure password, the purposes of these results are to show how many passwords are nearby that are leaked and unhashed with either the same name or any other names.

### 4.1 Hotmail Database



*Figure 7: Result from Hotmail database of password “Georgi”*

We can clearly view that similar passwords to “Georgi” has been unhashed from this database even with such a small amount of passwords. I will now pull some data from the information box from Figure 4 and see what results I will get.



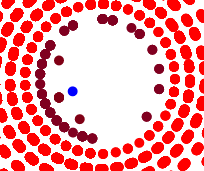
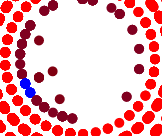
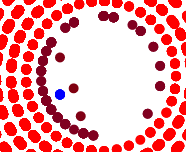
*Figure 7.1 Figure 7.2 Figure 7.3*

So if we look carefully, you can see a pattern where at the start you only have three dots at the most inner side, the results of those dots were “gloria”, “sergio”, “yerayi” by which two of these are clearly names. I wanted to go down the route of the dots below of the first dot Figure 7.1 which was “gloria” because its closest to the name “Georgi” and starts with the same letter so we get the benefit of the alphabetically sorted algorithm and we can see what the dots behind it were. Turns out majority of the passwords behind are also names of people for example Figure 7.2 we can see the passwords were “flores” and “gordis” and one can argue if those can be names but behind Figure 7.3 we have passwords such as “Garcia”, “felipa”, “felisa”, “Ferrero”, “felipe” which are much more clearly names of people. Considering this is a database of random passwords that were leaked from a website and still get matches of these names shows a pattern that people tend to use person’s names as passwords which is not ideal at all. The same test can be applied to objects and see if people are relating their passwords to objects.

### 4.2 MySpace Database

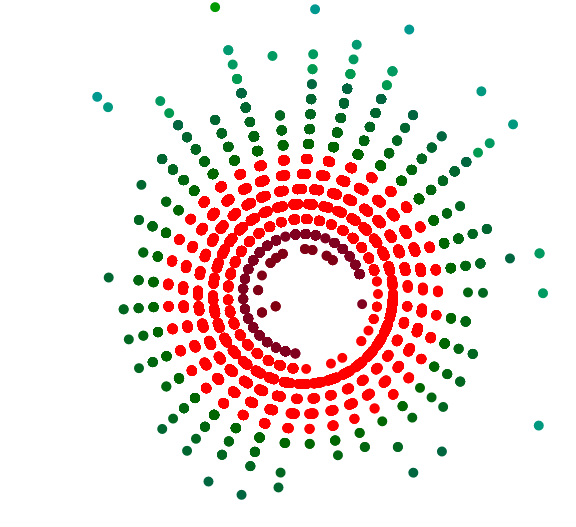
*Figure 8: Result from MySpace database of password “Georgi”*

We can see that this graph is quite a bit different but we will try to follow the same pattern by looking at the “G” area of the graph and see what we get now with much bigger database.

   
 *Figure 8.1 Figure 8.2 Figure 8.3*

Starting with the information from Figure 8.1, the password we get is “georgia”, following up with Figure 8.2 is “george3”, “georgia1”, “georgie6”, “george2” etc. which we now start seeing users using a clear pattern we talked about in Section 2.1 and are just adding on a number at the end of their password but at its core, its still their name adding a number than can be related to potentially their DoB or something else. Figure 8.3 seems to be a lot more similar passwords from Figure 8.2 but instead of adding single number, they add two numbers at the end proving that with the increase of database, we get a lot more people using these patterns of creating a password based on something easy they will remember and then adapting it with a simple pattern.

### 4.3 PHPBB Database



*Figure 9: Result from PHPBB database of password “Georgi”*

There is currently an issue with the selecting the dots when using the full database, the visualisation works but selecting the dots in a radius doesn’t seem to work because it array gets too big so instead we are using a database half its original size.

Repeating the same process, we are going to locate where the “G” character is and look down on the graph to see similarities.

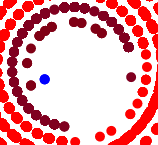
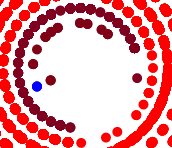
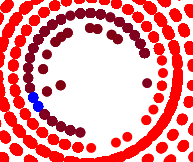


Figure 9.1 Figure 9.2 Figure 9.3

Starting with the initial Figure 9.1, we get the names of “georgie”, “georgia”, “george” with edit distance of 2. Moving onto next stage it has names such as “georgina”, “georges”, “giorgio” with edit distance of 3 and finally we have a bunch of names such as “gertie”, “gigi”, “george36”, “george10” etc. with edit distance of 4.

These results clearly show that no matter what the size of the database was, and the fact that it was gathered from completely different websites, the end results are all that you get a lot of users using their names and simple tricks to either create a new password or change their current one a bit to satisfy a password meter and believe that it will keep them secure from their accounts being entered to without permission.

## **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 exceeded on information box

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

## 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, me 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.