A Comparative Evaluation of the effectiveness of modern-day password policies

Jack Sime

CMP320: Ethical Hacking 3

BSc Ethical Hacking Year 3

2021/22

Note that Information contained in this document is for educational purposes.

.

Abstract

This paper looks at the current requirements set out by password policies on some of the most used websites around the world and subjects them to several common password cracking techniques in an attempt to evaluate just how effective the policies are. Password policies play a big part in account security and ensuring that the password policy being used on a website encourages the user to provide a strong and secure password can help with the overall safety of the user's account in the event that the password hashes for the website are stolen or leaked. Subjecting the different groups of password policies to a wide variety of tests can allow for any weaknesses and strengths of the policy to be identified from test to test and can be used to determine if a policy is outdated and should be amended to encourage the use of stronger and more varied passwords.

Ten of the most visited websites were selected and their password policies were noted and then group into five groups, each with different requirements. One hundred passwords were then created for each group that met the requirements and offered different combinations e.g. human-created, and password manager created. Each group was then hashed using a simple python program using both MD5 and bcrypt. The hashes were then subject to four different tests using web-based hash crackers, John the ripper with two different word lists and finally the use of rainbow tables using RainbowCrack, and the results were stored within an excel file for each group. A maximum cracking percentage was then calculated for each group and displayed the total percentage of different passwords that collectively would have been cracked if all four methods were used on the list of hashes during a proper attempt to crack them.

It was found that each policy demonstrated strengths and weaknesses against the attacks shown and that several of the policies that were used within the testing should be amended to improve the overall security. It was also seen that as the requirements for the password grew in variety the less effective the tests were against them. Overall key conclusions could be drawn in relation to both the effectiveness of each policy and where potential amendments would be recommended to bring the policy in line with the current cracking potential possessed by modern computers. It was also noted that the hashing algorithm used by the website can add security to the overall password hash even if the policy in use is not as effective as others in use. The policies used within this paper were able to be ranked based on the lowest overall cracking potential that was seen during the testing.

.

Contents

.

1 INTRODUCTION

1.1 BACKGROUND

At a security conference in 2004, Bill Gates said "There is no doubt that over time, people are going to rely less and less on passwords" (Kotadia, 2004) and since then there have been many changes made to password security including the widespread use of 2FA (Two-factor Authentication) in conjunction with passwords along with security tokens which are physical items required to successfully login (*What Is a Security Token? | Okta UK*, no date). Although these additions have been made, passwords continue to be used in connection with these rather than being replaced by them and so the traditional issues (Crafford, 2020) with passwords continue to exist and can be exploited. Passwords are required for nearly everything that can be done in the modern world whether it be at work or home, you're always having to create passwords for things as well as remember them. A survey conducted by Google and Harris Poll in 2019 (Figure 1) revealed that from a group of 3000 adults in America 52% reuse multiple passwords over accounts and 13% use the same password for all their accounts (*Google & Harris Poll Survey*, 2019).

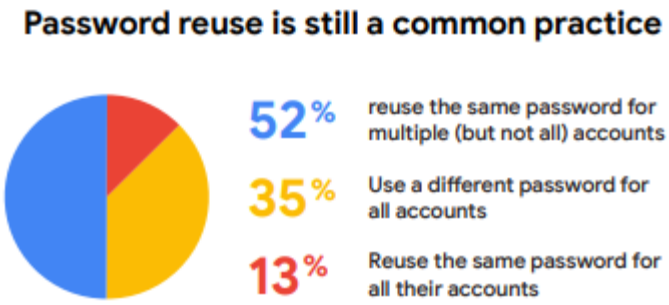
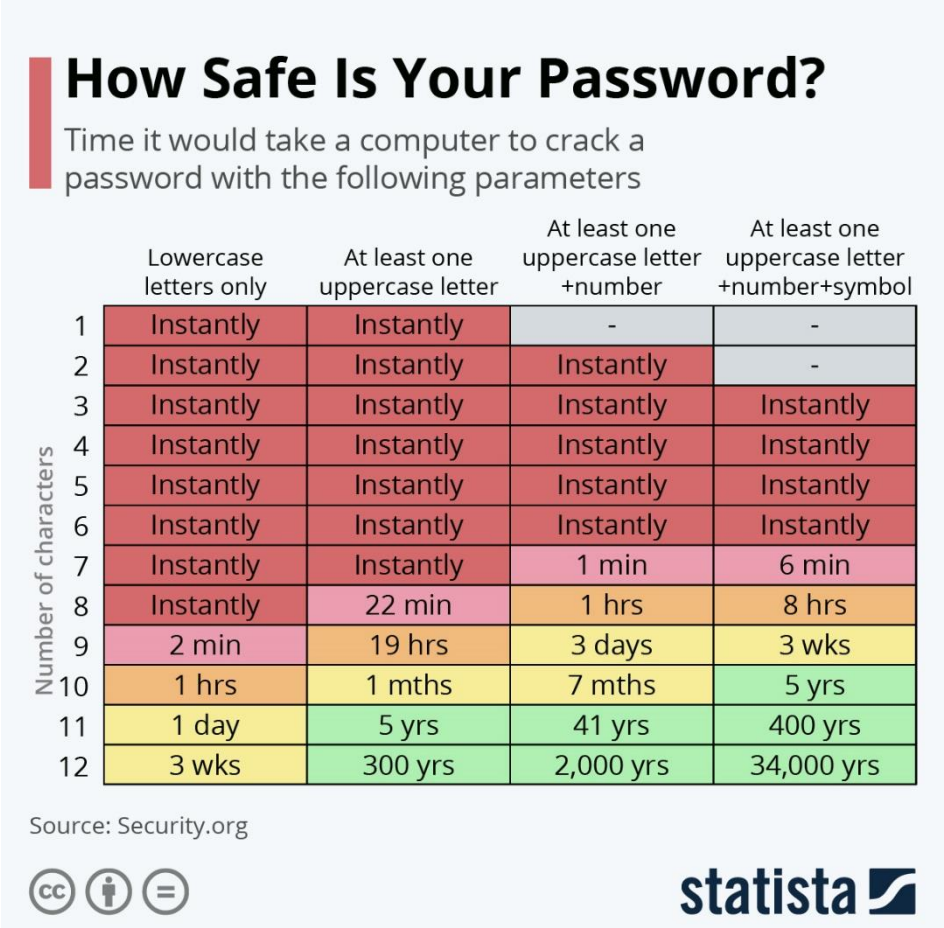


Figure 1: Password Reuse from the study

Many different websites deploy a wide range of password policy variations including minimum and maximum requirements along with some combination of letters and/or numbers and symbols, in the hope that these policies help to prevent an account from being stolen or hijacked. Ensuring that passwords are both diverse and unique over different accounts can help in the event that an account gets breached, and the enforcement of good policies can help facilitate this. After submitting your password to a website to create an account, your password should be hashed using one of many algorithms available to obfuscate the true nature of the submitted value and help prevent it from being used if the website database is breached. Algorithms such as MD5 and bcrypt (*PHP: password_hash - Manual*, no date) are used to hash passwords on websites but with the developments in modern technology allowing for passwords to be stolen and cracked with increasing speed and efficiency, ensuring that websites encourage good password creation habits can provide benefits for the website and the user in ensuring that accounts are kept as safe as possible (Figure 2) (*Infographic: How Safe Is Your Password?*, no date). With MD5 still being heavily

used within website settings and having been proven to be insecure and have collisions (Ramirez, 2015, p. 5), there are various tools and websites available that can crack passwords hashed using MD5 so having unique and diverse passwords can reduce the chance, should your password hash be leaked, that it will be cracked quickly.



How Safe Is Your Password?

Time it would take a computer to crack a password with the following parameters

Number of characters	Lowercase letters only	At least one uppercase letter	At least one uppercase letter +number	At least one uppercase letter +number+symbol
1	Instantly	Instantly	-	-
2	Instantly	Instantly	Instantly	-
3	Instantly	Instantly	Instantly	Instantly
4	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	Instantly
7	Instantly	Instantly	1 min	6 min
8	Instantly	22 min	1 hrs	8 hrs
9	2 min	19 hrs	3 days	3 wks
10	1 hrs	1 mths	7 mths	5 yrs
11	1 day	5 yrs	41 yrs	400 yrs
12	3 wks	300 yrs	2,000 yrs	34,000 yrs

Source: Security.org

statista

Figure 2: Brute force statistics

Ensuring the minimum password requirements detailed in password policies being deployed on websites containing particularly sensitive data, such as social media sites and websites that relate to potential income sources such as PayPal and eBay, is important in ensuring that you're taking the best steps towards securing your accounts in the best way possible. Although almost every site has a password policy in place, many are insufficient or outdated given the capabilities of potential attackers and so a general password policy that should be followed regardless of the requirements that are set by the website will be best in protecting your account.

1.2 AIM

This report aims to test and evaluate the effectiveness of modern-day password policies in use within websites. This overall aim comprises of several sub aims:

- Recreate the hashing environment that modern websites would utilize
- Create a large selection of realistic password examples to be tested
- Successfully test the policies using a range of hash cracking techniques
- Evaluate the effectiveness of each policy included in the test
- Discuss the implications of each password policy and its strength
- Provide a comparison of the different policies and provide corrective measures to improve password security

2 PROCEDURE

2.1 OVERVIEW OF PROCEDURE

During the creation of the passwords for the groups, the absolute minimum that the policy required was achieved to ensure accurate and proper testing of the true strength of the policy. Although this is unlikely to represent the average user's password, the true strength of the policy had to be properly evaluated. For the purpose of the testing and creation of the methods used below, a consumer-grade computer was used. The specs of the used system included an I7-9700K, RTX2070 super, and 32Gb of RAM.

2.2 PASSWORD POLICY RESEARCH

To begin with, research was conducted to identify ten largely used websites around the world that would be of interest to a potential attacker. Viewing a report from Visual Capitalist (Neufeld, 2021), the top fifty were identified and ten were selected that met the criteria of providing either personal information or access to some monetary value such as banking details or card numbers. The ten websites selected are listed below:

1. Google (YouTube, Gmail, etc)
2. Twitter
3. Microsoft
4. Amazon
5. Netflix
6. Facebook
7. Instagram
8. eBay
9. PayPal
10. Twitch

After the websites had been chosen, the current password policies being used on the websites had to be investigated and noted down. Using online tools such as Passhints.com (*Password hints for websites and mobile apps*, no date) and passrequirements.com (*Password requirements | PassRequirements.com*, no date) along with manual testing of the requirements, the password criteria for each of the websites was gathered. Websites were then grouped based on their password requirements and these formed the testing groups for evaluation. The groups were set up as follows:

Group 1 (Minimum 8 only)-

- Google
- Twitter
- Microsoft- has weak password protection in place

Group 2 (Minimum 6 only)-

- Amazon
- Netflix
- Facebook- has weak password protection in place

Group 3 (Minimum 6 with Number and Symbol)-

- Instagram- has weak password protection in place
- eBay

Group 4 (Minimum 7 with number and symbol no spaces)-

- PayPal

Group 5 (Minimum 8 with Capital and Number)-

- Twitch- has weak password protection in place

With the groups created and the criteria for each website known, the next step was to generate the password list for each group that will be hashed and then used to evaluate the policy being used. Each group would be tested using a set of one-hundred passwords which would be split 60:40 with 60% of the list being made up of passwords that a large amount of the population would create e.g common words contained within the dictionary etc, with the remaining 40% being made up of passwords more akin to that of password managers or security smart individuals. A table detailing the full split can be seen in Appendix A.

A wordlist of 58k words in size was used (*The Corncob list of more than 58 000 English words*, no date) to select the bulk word to be used within the sixty percent password split and the tester included the additional requirements where necessary. The forty percent split was created using SecureSAFE Pro Password Generator (*Free Password Generator*, no date) and no alterations were made to the passwords produced using the software. Each group had its own list of passwords to be used for testing. The complete lists of passwords used can be seen in Appendix B.

2.3 PASSWORD HASHING

After creating the lists, the passwords had to be hashed for the testing to begin. For hashing the passwords two different algorithms were used that would be used within a website environment. Md5 was selected as it is still currently in use within websites, even though it has been proven to be unsafe, due to the work required to overhaul the process and implement a newer algorithm (Cimpanu, 2019, p. 5). The other algorithm that was selected was bcrypt (Arias, 2021) as it has been adopted by PHP from version 5.5 onwards to be the default hashing algorithm when using the "password_hash" function and sets a standard for future website hashing. A python program was created that would open the selected list of passwords to be hashed along with opening a file for both the MD5 and the bcrypt hash. The program would then hash the password and write the hashes to one of the two output files depending on the algorithm used. This program allowed for the large number of passwords to be efficiently hashed. Passwords were read from the list where they were stored and were hashed using the "hashlib" module (*hashlib — Secure hashes and message digests — Python 3.10.4 documentation*, no date) for md5 and the "bcrypt" module for bcrypt (*bcrypt · PyPI*, no date). The original string was written first and then the hashed value was then written out to the newly created file to store the hashes (Figure 3). The program used can be seen in Appendix C.

```
DCBrreee:  Salt : $2b$12$4mlF748EMStIBZPlt5K2Vu Hash : $2b$12$4mlF748EMStIBZPlt5K2VunNPiXnfdPvL6cNHFiLgSc9X6WblpFI6
HhZlPJTk:  Salt : $2b$12$cVlvwEDECdQAUcWyTHCgq. Hash : $2b$12$cVlvwEDECdQAUcWyTHCgq.ivttAZa9UvuWpAjktrhdqs.EA4Jul.O
DBBxtwoh:  Salt : $2b$12$tOha6nUPZlfgfjAQbCZi4e Hash : $2b$12$tOha6nUPZlfgfjAQbCZi4eAty29UzBomiNgUfcLrKjLulUMsSgCVu
OMymtsvg:  Salt : $2b$12$HPgbpn9tDOkeOpH1RAbAru Hash : $2b$12$HPgbpn9tDOkeOpH1RAbAruB6oFriGs2ENW9GYvwqMJUp03BmUEEtG
YKAyqzSn:  Salt : $2b$12$twrtyrspkUFef72N1iW2uu Hash : $2b$12$twrtyrspkUFef72N1iW2uuyOjvJYBf.f8zUsr5TMpHvQPNME40PS6
KWNmghqM:  Salt : $2b$12$icTS45IQbZS0TMv7PkKQN. Hash : $2b$12$icTS45IQbZS0TMv7PkKQN.IOcAHvhJfYOvxeGhvnTdWw4rvzESp7u
sKAcuyex:  Salt : $2b$12$iQ9ZuFTWEhTCxlrBCZeLnu Hash : $2b$12$iQ9ZuFTWEhTCxlrBCZeLnuzYyP4ucgdzPMu3WmSQd2azwqE6RRKg.
NwjqxCap:  Salt : $2b$12$IN48cIdLVvHn5J50EyuYy. Hash : $2b$12$IN48cIdLVvHn5J50EyuYy.mIdin9mbUiHJz/ATMSF60YGJvTY5Kcu
RlskXfdJ:  Salt : $2b$12$0iGVDjWniE/m2oig8QzpgO Hash : $2b$12$0iGVDjWniE/m2oig8QzpgOUiEAlqQrt.bKzNbbuIcmY5aguNSRqA.
opNFNOhi:  Salt : $2b$12$o5OpCvd502hpCkdMM/cRce Hash : $2b$12$o5OpCvd502hpCkdMM/cRceDTR0LuScB6pN.KvLpPEgvnuE/fPsNKW
```

Figure 3: Example file output from hashing program

2.4 TESTING

After successfully creating the password groups and hashing them, the passwords were then tested using a variety of different password cracking techniques ranging from easy to access websites to software specially designed to crack passwords. After each test, the percentage of passwords cracked was calculated and was used to make comparisons in later sections. Results were tracked for each group using individual excel files as this would allow for results to be easily sorted and displayed. For each test, an average was calculated based on the number of passwords that were cracked and was used in the evaluation of the policy.

2.4.1 WEBSITES

With a wide range of password cracking websites available, they offer a quick and easily accessible option to crack a wide range of different password hashes. The website used when cracking the hashes was "Hashes.com" (*Decrypt hashes for free online Hashes.com*, no date) as it stated on the website that it supported both algorithms that were being used in the testing. The passwords from each group were tested in batches of twenty-five due to limitations on the website. The batches of twenty-five were submitted to the website and the results were then noted within each group's excel file. After testing

every password from each group, a percentage was calculated that represented the effectiveness of the password policy for that group. This percentage would form the basis for the comparison of the groups.

2.4.2 JOHN THE RIPPER

After performing testing using an online web-based hash cracker the software John the Ripper was used (*John the Ripper password cracker*, no date). John the ripper was specifically used with two different wordlists that differed in size and availability. Each John the ripper run was left to run to competition and there was no time limit set on the cracking time for any groups. The two wordlists used were rockyou.txt (*Common Password List (rockyou.txt)*, no date), which is a commonly used wordlist and is readily available within Kali Linux which is the system that was used to run JTR, and Md5decrypt-awesome-wordlist (*Weakpass*, no date), which is a large wordlist with upwards of one billion entries in it and available from an online password cracking website among other sources. As previously mentioned, a Kali Linux VM was used to complete the JTR operations and the results from these tests were also stored within the excel document for each group of passwords. Both wordlists were tested using the same JTR command seen in Figure 4.

Figure 4: JTR command used

After the cracking had finished, after each group another command was used to access the complete list of passwords cracked for that given group. This can be seen in Figure 5.

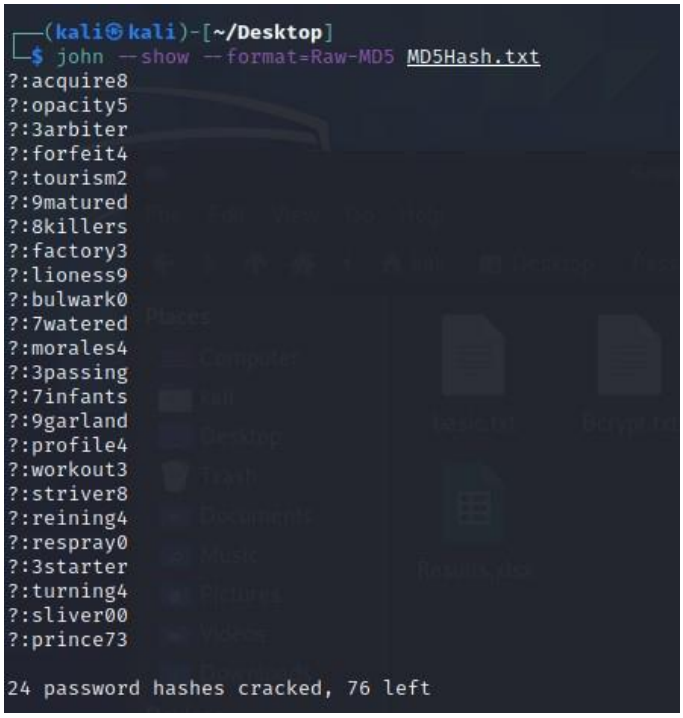


Figure 5: JTR show command used after each individual group

2.4.2.1 ROCKYOU.TXT

The first round of cracking utilized the previously mentioned “rockyou.txt” as the wordlist that was supplied to JTR. The first group of passwords had their list of MD5 hashes, which had been previously created, loaded into JTR, and then the application was then started. The application was left to run until completion and once completed, the “show” command was used to reliably reveal the correct number and value of the passwords that had been cracked. After obtaining the correct list of cracked passwords the results were then recorded within the excel document for the current group being tested. The percentage of passwords cracked was then calculated and recorded within the excel document. Whilst using the rockyou.txt wordlist each crack attempt took several minutes from start to completion. This process was then repeated for each of the five groups of passwords but was limited to the MD5 set of hashes only. This was due to how the bcrypt algorithm was designed along with the computational power required to calculate the number of hashes required and the overall time that would have been needed while using the current wordlist.

2.4.2.2 MD5DECRYPT-AWESOME-WORDLIST

After the cracking had been finished using the “rockyou.txt” wordlist, the wordlist being used was then swapped to the larger “Md5decrypt-awesome-wordlist” and the first group of password hashes was then loaded into JTR again and the application was started. The same steps that were taken when using the “rockyou.txt” wordlist were then repeated along with the command that was used to start JTR was replicated but with the different wordlist loaded instead. Whilst using this wordlist, each group took on average ninety minutes to run from start to completion and after JTR was completed the same steps used previously were followed once again to accurately record the results within the excel file and calculate the percentage for each group. The bcrypt variation of the password’s hashes had to be skipped once again due to the previously mentioned limitations of the testing environment and the overall construction of the algorithm.

2.4.3 RAINBOWCRACK

After using both website-based password crackers and wordlist-based crackers, the last method that was used was rainbow tables. For this, the application RainbowCrack was used, and using the additional software provided with RainbowCrack each group had its own set of rainbow tables created that met the requirements of the password policy for that group. Rainbow tables can generally be over several hundreds of gigabytes big (*List of Rainbow Tables*, 2020) but due to the resources required to create tables of that size the tables used within the testing were 2.2GB in size. Each group had its own custom charset used when creating the rainbow tables which can be seen in Figure 6.

```
group-5   = [abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789]
group-4   = [abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*()-_+=~`[]{}|\:;"'<>,.?/]
group-3   = [abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*()-_+=~`[]{}|\:;"'<>,.?/ ]
group-2   = [abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ ]
group-1   = [abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ ]
```

Figure 6: Charsets for rainbow tables

Due to the way that the bcrypt algorithm works, rainbow tables are too expensive in resource terms to be used to crack hashes of that nature.

After setting up the correct charsets for the groups, the rainbow tables were then ready to be created. Each group had 6 tables created and took about two hours to create the six tables for one group, these tables used the same command but with the table index ranging from 0-5 hence the six tables per group. The table index is responsible for selecting the reduction function which trims combinations that would be seen as extremely unlikely, creating multiple tables with ranging indexes allows for near-perfect tables to be created. Included with RainbowCrack was the application “rtgen.exe” which was used to create the tables and the same set of commands was used throughout testing and can be seen in Figure 7.

```
rtgen md5 group-1 1 8 5 2400 24652134 0
rtgen md5 group-1 1 8 4 2400 24652134 0
rtgen md5 group-1 1 8 3 2400 24652134 0
rtgen md5 group-1 1 8 2 2400 24652134 0
rtgen md5 group-1 1 8 1 2400 24652134 0
rtgen md5 group-1 1 8 0 2400 24652134 0
```

Figure 7: commands used to create the tables for group 1

Due to the length of time required to compute all the required tables, a batch file was created that created all thirty of the required tables. This file can be seen in Appendix D. After each group had their tables created the tables had to then be sorted, so that they could be utilized, using “rtsort.exe” which was also included with RainbowCrack (*Rainbow Table Generation and Sort*, 2020). Once the tables had been sorted each group of tables was then merged using “rtmerge.exe” and were now ready to be used.

For the physical testing, the RainbowCrack software “rcrack_gui” was used to load in each separate list of hashes and then load in the group of tables designed for that set of hashes. The application then attempted to crack the hashes and after on average one minute the results were returned along with a list of cracked hashes. This process was repeated for each of the five groups and the results were recorded within each group's excel spreadsheet. This was the final testing that was attempted on the groups of hashes.

3 RESULTS

Each group's results will be looked at separately and along with the results of the four different tests, a combined result will be calculated to truly represent the strength of the policies if they were to be subject to the four tests in a combined effort to crack the hashes. Unfortunately, due to several factors, none of the bcrypt versions of hashes were able to be cracked within the testing. A complete table of results can be seen in Appendix E.

3.1 RESULTS FOR GROUP 1

Group 1's requirements were eight letters and featured both uppercase and lower case along with some spaces. The results from the testing were largely similar with group 1 over the four different tests.

Beginning with the initial testing using web-based crackers, 30% of the one hundred passwords were able to be cracked. The complete 30% cracked were singular words and didn't feature a two-word combination or any of the randomly generated passwords like the ones produced by modern-day password managers.

Moving onto the use of JTR and the two wordlists selected, the results were largely like that of the websites. Starting with the "rockyou.txt" wordlist, the results showed that a total of 29% of the passwords had been cracked using this word list. As with the websites, the only passwords that were cracked were complete words and none of the two-word combinations or the randomized passwords had been cracked. The only complete word that was not cracked by this wordlist was the word "manually" and this was the only change in results.

Looking at the use of the larger wordlist "Md5decrypt-awesome-wordlist", a total of 30% of the passwords were once again cracked. The passwords cracked using this list were identical to that of the ones cracked using the website-based crackers. A trend of only the complete words being cracked was evident at this point of the testing for group 1.

The final test using rainbow tables returned a large difference in results as only 1% of the passwords were able to be cracked through this method. As with the other tests, it was a complete word that was cracked during this test. The overall impact of the rainbow tables test will be discussed in a future section.

Combining the four different tests, a maximum possible cracking percentage for group 1 was calculated and was equal to 30% of the total passwords.

3.2 RESULTS FOR GROUP 2

Group 2's requirements were eight letters and featured both uppercase and lower case along with some with spaces. The results from the four tests featured more variety but also still included large areas of commonality.

Starting with the web-based cracking results, a total of 33% of passwords were successfully cracked. Once again, the complete group of singular words were cracked along with two of the two-word combinations and one of the randomly generated passwords.

Using the "rockyou.txt" wordlist with JTR resulted in a total of 22% of the passwords being able to be successfully cracked. The passwords successfully cracked were limited to complete words again and none of the other password variations were able to be cracked. Several complete words were also left uncracked after using this word list.

Switching to the Md5decrypt wordlist also used revealed a total of 33% of passwords being cracked. As with the web-based crackers, every password that utilized only a single word was cracked and three of the two-word combinations were cracked whilst zero of the randomly generated ones was cracked. As with group 1, a common theme is easily identified at this stage of the group 2 testing.

Moving onto the final test utilizing rainbow tables, an outlying result was achieved and 100% of the password hashes were successfully cracked using the generated tables. Compared to the previous tests this was a massive improvement in the success of cracking.

After combining the results of the four tests, the complete list of passwords had been successfully cracked and so a maximum possible cracking percentage was calculated at 100% of the total passwords hashed.

3.3 RESULTS FOR GROUP 3

Group 3's requirements were six letters, numbers, and symbols along with featuring both uppercase and lower case along with the use of spaces. The results from the four tests featured more variety but also still included large areas of commonality.

The first test using the website password crackers returned a result of 6% of the passwords being successfully cracked. All the passwords cracked in this test featured a full word contained within the password along with a symbol and a number.

The results achieved from using both wordlists with JTR were a total of 0% cracked for either list. Neither list was able to successfully crack a single password that had been created using the requirements set out for group 3.

Looking at usage of the rainbow tables for group 3 returned a result of 42% of all passwords being successfully cracked. The rainbow tables managed to crack a wide range of different password creation combinations and the result was a massive increase in the potential over the other three tests conducted.

Combining the 4 test results, the maximum possible cracking percentage was 44% of the total passwords. 42% was from the rainbow tables and the additional 2% had come from passwords that the rainbow tables failed to crack but had been cracked using the web-based crackers.

3.4 RESULTS FOR GROUP 4

Group 4's requirements were seven letters, numbers, and symbols along with featuring both uppercase and lower case with the exclusion of the space character. The results from the four tests were extremely similar in their outcome.

From the testing results, it was found that group 4 was the hardest set of passwords to be cracked so far. The results from the web-based testing showed that only 1% of passwords had been possible to crack. The cracked password was one that featured a full word in it.

As was with group 3 the usage of both different wordlists with JTR returned a total of 0% of passwords successfully cracked. Once again both wordlists were unable to successfully crack one of the hashed passwords.

The results from the rainbow tables for group 4 were like that of the web-based cracking where only 1% of passwords were successfully cracked. Unlike the web-based cracking though the password cracked during the rainbow table testing was a randomly generated one that simulates a password manager-created password.

After combining the results from the four tests, a maximum potential cracking ability was revealed to be 2% of the passwords tested for group 4.

3.5 RESULTS FOR GROUP 5

Group 5's requirements were eight letters along with numbers and featuring both uppercase and lower case. The results from the four tests were largely widespread in their cracking potential.

The web-based hash cracker was able to uncover a total of 40% of the passwords that were used within group 5. Every password cracked contained a solid word within it and none of the randomly created passwords were successfully cracked with this method.

The "rockyou.txt" wordlist was able to successfully crack a total of 24% of the passwords but was also only able to crack passwords that contained a full word within them. There were some levels of overlap between the first two tests and the password that had been successfully cracked.

Moving onto the much larger wordlist used with JTR a total of 39% of the passwords submitted were successfully cracked. A similar trend was seen between the first three tests where only passwords containing a full word were able to be successfully cracked. There was now a large area of overlap after the first three tests had been concluded.

The rainbow tables results were limited to 1% of the passwords being cracked using this method. Unlike the previous three tests, the singular password that was successfully cracked was a randomly generated password and did not contain any full words within it.

After looking at the results from all four tests, a combined maximum cracking potential was seen to be 44% with the main quality coming from the results achieved using JTR.

4 DISCUSSION

4.1 GENERAL DISCUSSION

Before discussing the overall results of the testing, the use of rainbow tables should be discussed and explained. The rainbow tables that were created for each group were 2.2GB in size per group. These tables are massively undersized compared to the tables that would realistically be used when conducting the process of hash cracking with rainbow tables. Although two out of the five groups returned very positive results using rainbow tables, the overall effectiveness of tables was not fully displayed within the testing and so with larger tables, there's a high probability that a higher percentage of passwords could be cracked.

From the results, it can be seen that as the policy gets more complicated, with the addition of longer minimum requirements and further requirements of symbols etc, the overall effectiveness of the cracking methods used decreases (*Estimating Password Cracking Times*, no date). Both wordlists were returning results within 20%-30% on policies that utilized only letters and numbers but could only manage <5% returns on the policies that had extended requirements. This overall drop highlights the large benefits that can be achieved with small additions and tweaks to existing policies that are being used.

This trend continued for the majority of the testing, where the policy variety was increased either by length or by the minimum requirements the overall effectiveness of the cracking methods as a whole was decreased. Outliers within the testing did appear such as the rainbow tables result for group 3 where 42% of passwords were cracked. This result along with several others seen within the testing highlight the effectiveness and benefits of utilizing multiple attack directions along with demonstrating the strengths of the policies against these different attacks.

From the results of the testing, it can be noted that the use of an algorithm when hashing the passwords by the website can play a big part in the effectiveness of the website's security. Even if the website was using a weaker policy such as the policy used by group 2, ensuring that the website had a strong hashing algorithm in place would add security. As seen within this testing with the use of bcrypt, anyone attempting to crack the password hashes of a strong algorithm such as bcrypt, using consumer-grade hardware, is going to be forced to spend a large quantity of time attempting to crack them compared to the same policy but using a weaker algorithm such as MD5 ('How secure is bcrypt? | synkre.com', 2019).

Looking at overall results in relation to the attack methods used allows for the weaknesses and strengths of each policy to be identified. Group 1 scored near 30% on every test except for the rainbow tables where it scored 1%. As previously mentioned, the size of the tables used impacted the performance of them and with larger tables, the results for group 1 would have been higher. Of that 30%, it was seen that every password contained a proper word so adopting the use of a randomized password created by software etc would nullify the results that were produced.

Group 2 scored from mid-20% to low 30% across all tests except the rainbow tables again. The size of the tables didn't impact their performance for group 2 as 100% of the passwords were able to be cracked. This individual result highlighted the true weaknesses of the group 2 policy and the shortfalls that may appear if a user was to create a password that met the minimum specs of the group.

Group 3 scored <10% in the first three tests and then scored 42% in the rainbow test. Once again the small tables proved to be effective against the requirements of the group 3 policy and highlighted the potential for the policy to be weak to this type of attack. It also highlighted the strengths of the policy against the previous three different attacks having limited the cracking potential to a max of 6% on the first three tests.

Group 4 scored <2% in all four of the tests that were conducted and had a total of 2% of the maximum cracking potential. The results displayed the strengths of the policy, against the variety of different tests conducted, limiting the testing to minimal results. The multiple requirements set within the policy allow for possible password combinations to be greatly increased over other policies with less specific requirements.

Group 5 had varying scores that range from 1% up to 40% and the scores varied with each of the different tests. The results from tests one and three, where web-based crackers and the Md5decrypt-awesome-wordlist was used with JTR, both returned results within the 39%-40% showing the possible weaknesses against these tests. The results from the "rockyou.txt" wordlist also scored 24% and displayed more potential weaknesses of the policy to attacks utilizing wordlists.

From the results gathered the groups can be ranked from best to worst where 1 is the best and 5 is the worst as seen below:

1. Group 4
2. Group 3
3. Group 1
4. Group 5
5. Group 2

4.2 CONCLUSIONS

From the findings displayed within this paper, several key conclusions can be drawn in relation to both the password policies along with the testing methods that were used. From the results, it should be noted that although the policy being used plays a big part in the security of a user's account, the handing of the user's password in terms of hashing and storage also plays a big part as was seen with the usage of bcrypt and the lack of options to successfully crack the passwords hashed using that algorithm.

All four of the testing methods used within the paper were available to the public and could be carried out by anyone with a suitable computer. It can be stated that with more professional hardware and more resources, a higher percentage of the passwords used within each group would have certainly been cracked (Goodin, 2012).

From the results, groups 1,2, and 5 should amend their password policies in some manner to ensure that the minimum requirements for the websites contained within these groups are increased to encourage users to use more suitable and secure passwords. Adding the need for symbols along with increasing the minimum password length would increase the security of the passwords created by the site by a large margin. Websites contained within these groups such as Google and Twitch can heavily relate to a user's income and so having out of date policies being used to protect valuable user information, along with potentially pertaining to a user's career (Playbook, no date), displays how important a role password policies can play within the security of the website and its users.

Over the results for all five groups, the passwords that contained at least one solid word used within the English language were almost certainly cracked first and sometimes were the only ones cracked. When creating a password, single words should be avoided as seen within the testing they are highly susceptible to being cracked. Instead, either a random password generated by a password manager should be used or a passphrase should be used where multiple words are combined to create a longer phrase (*Use a Passphrase*, no date) that is much less susceptible to being cracked easily.

Overall, from the results gathered and the patterns observed during testing a generalized password policy that should always be followed where possible was able to be created. Ensuring that any passwords created are eight characters in length at a minimum is an important starting point as passwords of less length have fewer combinations and so are easier to crack. Ensuring that along with a minimum of eight characters in length, a variety of characters are used such as the use of both upper and lower case and the inclusion of symbols, numbers, and spaces where possible. Lastly, when creating passwords avoid using a single full word as the main body of the password, try either creating a random password using a tool or software or create a passphrase where the password consists of several words used in tandem to create a longer phrase. When creating a passphrase using random words or words of items within the user vicinity, such as "park cat gate", along with the previously mentioned tips will provide the user with a safe and secure password that will be hard to crack if it's ever stolen (*Creating Strong Password Policy Best Practices | DigiCert.com*, 2014).

4.3 FUTURE WORK

In relation to future work on this subject, several areas that could be expanded on have been discovered. Obtaining pre-calculated rainbow tables of significant size (100GB+) would allow for a better understanding of how the password policies would stand up to an attack that utilized them. It would also provide a more realistic test of how the policies would hold up if the password hashes for a certain website were leaked and attempted to be cracked.

Sourcing specialized hardware designed to be used with password cracking etc would allow for more tests to be conducted along with other tests such as brute-forcing which relies on fast equipment to be effective (*Password Cracking Rig Guide | White Oak Cyber Security*, 2019).

Incorporating an area of public involvement would allow for more realistic passwords to be used within the testing. Having members of the public create passwords that met the requirements for the groups would incorporate a more realistic evaluation as the passwords would truly represent the hashes that may be stolen during an attack and would be free from any potential bias that comes from a single creator (*Most hacked passwords revealed as UK cyber survey exposes gaps in online security*, 2019).

REFERENCES

Arias, D. (2021) *Hashing in Action: Understanding bcrypt*, *Auth0 - Blog*. Available at: https://auth0.com/blog/hashing-in-action-understanding-bcrypt/ (Accessed: 24 May 2022).

bcrypt · PyPI (no date). Available at: https://pypi.org/project/bcrypt/ (Accessed: 24 May 2022).

Cimpanu, C. (2019) *A quarter of major CMSs use outdated MD5 as the default password hashing scheme*, *ZDNet*. Available at: https://www.zdnet.com/article/a-quarter-of-major-cmss-use-outdated-md5-as-the-default-password-hashing-scheme/ (Accessed: 21 May 2022).

Common Password List (rockyou.txt) (no date). Available at: https://www.kaggle.com/wjburns/common-password-list-rockyoutxt (Accessed: 21 May 2022).

Crafford, L. (2020) *The State of the Password Problem in 2020*, *The LastPass Blog*. Available at: https://blog.lastpass.com/2020/12/the-state-of-the-password-problem-in-2020/ (Accessed: 11 April 2022).

Creating Strong Password Policy Best Practices | DigiCert.com (2014). Available at: https://www.digicert.com/blog/creating-password-policy-best-practices (Accessed: 24 May 2022).

Decrypt hashes for free online Hashes.com (no date). Available at: https://hashes.com/en/decrypt/hash (Accessed: 11 April 2022).

Estimating Password Cracking Times (no date) *Better Buys*. Available at: https://www.betterbuys.com/estimating-password-cracking-times/ (Accessed: 24 May 2022).

Free Password Generator (no date). Available at: https://www.securesafepro.com/pasgen.html (Accessed: 11 April 2022).

Goodin, D. (2012) *25-GPU cluster cracks every standard Windows password in <6 hours*, *Ars Technica*. Available at: https://arstechnica.com/information-technology/2012/12/25-gpu-cluster-cracks-every-standard-windows-password-in-6-hours/ (Accessed: 24 May 2022).

Google & Harris Poll Survey (2019) *Online security Survey*. Available at: https://services.google.com/fh/files/blogs/google_security_infographic.pdf (Accessed: 11 April 2022).

hashlib — Secure hashes and message digests — Python 3.10.4 documentation (no date). Available at: https://docs.python.org/3/library/hashlib.html (Accessed: 24 May 2022).

'How secure is bcrypt? | synkre.com' (2019), 14 December. Available at: https://synkre.com/how-secure-is-bcrypt/ (Accessed: 24 May 2022).

Infographic: How Safe Is Your Password? (no date) *Statista Infographics*. Available at: https://www.statista.com/chart/26298/time-it-would-take-a-computer-to-crack-a-password/ (Accessed: 15 May 2022).

John the Ripper password cracker (no date). Available at: https://www.openwall.com/john/ (Accessed: 21 May 2022).

Kotadia, M. (2004) *Gates predicts death of the password*, *CNET*. Available at: https://www.cnet.com/news/privacy/gates-predicts-death-of-the-password/ (Accessed: 11 April 2022).

List of Rainbow Tables (2020). Available at: http://project-rainbowcrack.com/table.htm (Accessed: 24 May 2022).

Most hacked passwords revealed as UK cyber survey exposes gaps in online security (2019). Available at: https://www.ncsc.gov.uk/news/most-hacked-passwords-revealed-as-uk-cyber-survey-exposes-gaps-in-online-security (Accessed: 24 May 2022).

Neufeld, D. (2021) *Ranked: The 50 Most Visited Websites in the World*. Available at: https://www.visualcapitalist.com/the-50-most-visited-websites-in-the-world/ (Accessed: 11 April 2022).

Password Cracking Rig Guide | White Oak Cyber Security (2019). Available at: https://www.whiteoaksecurity.com/blog/password-cracking-rig/ (Accessed: 24 May 2022).

Password hints for websites and mobile apps (no date) *PassHints*. Available at: https://passhints.co/ (Accessed: 11 April 2022).

Password requirements | PassRequirements.com (no date). Available at: https://passrequirements.com/ (Accessed: 11 April 2022).

PHP: password_hash - Manual (no date). Available at: https://www.php.net/manual/en/function.password-hash.php (Accessed: 15 May 2022).

Playbook, A.S. (no date) 'Is Streaming A Good Career? Things To Consider – Streamers Playbook'. Available at: https://streamersplaybook.com/is-streaming-a-good-career/ (Accessed: 24 May 2022).

Rainbow Table Generation and Sort (2020). Available at: http://project-rainbowcrack.com/generate.htm (Accessed: 24 May 2022).

Ramirez, G. (2015) *MD5: The broken algorithm*, *Avira Blog*. Available at: https://www.avira.com/en/blog/md5-the-broken-algorithm (Accessed: 15 May 2022).

The Corncob list of more than 58 000 English words (no date). Available at: http://www.mieliestronk.com/wordlist.html (Accessed: 11 April 2022).

Use a Passphrase (no date). Available at: https://www.useapassphrase.com/ (Accessed: 24 May 2022).

Weakpass (no date). Available at: https://weakpass.com/wordlist/1319 (Accessed: 21 May 2022).

What Is a Security Token? | Okta UK (no date). Available at: https://www.okta.com/uk/identity-101/security-token/ (Accessed: 11 April 2022).

APPENDICES

APPENDIX A – CREATION SPLIT

Basic	Complex
60%	40%

Basic refers to passwords that would be created by a human user using words within the English language.
Complex Refers to the passwords that could be created by a password manager/software as opposed to an actual person (no full words used)

APPENDIX B – LIST OF PASSWORDS

Group 1:

emission
purchase
messages
blocking
standard
peaceful
resolved
yourself
arrested
zimbabwe
nebraska
revision
painting
maintain
resolved
integral
retained
hometown
manually
japanese
programs
sampling
auckland
thriller

enormous
breeding
aviation
roulette
equality
routines
leak put
flag Paw
FREE peg
Also eth
Paint oo
Rex warn
lime ido
Deck jen
Pap stay
fuel Sid
HUGO hum
pin Pomp
Waddy av
Arm pasa
zip team
Bee Cold
left zoo
zoo eggs
Three or
cosy leg
Toe sons
a organs
bar vent
Ask hove
also mod
Sent egg
ice Gang
good ski
coal cup
by loyal
DCBrreee
HhZlPJTk
DBBxtwoh
OMymtsvg
YKAyqzSn
KWNmghqM
sKAcuyex
NwjqxCap

RlskXfdJ
opNFNQhj
VguQqqdc
wOoCOCKy
FwYlDHts
KokKHdmb
tusSHqQv
AArrpaej
qNkxVvZU
aQDrBasx
opmJoqAH
CLJRGMqd
rFPNCvqz
NuTgkbib
pGTulusp
rYrrsTYC
ZokQiQNc
oHUBHVMw
raUwIUYs
XXtGbzEe
KvqbhdYE
nxgHwAPs
tLgYdbvR
xlOkHpdK
kpNTGQpJ
yLOPjjYs
ILaXNRMs
sQvfMCDi
OKdmijxE
xFDwZcPu
NexKtFxF
gGzyMtrH

Group 2:

linkup
tremor
salons
ragbag
inlaid
jotted
defied
pilfer
scrums

tutors
verges
heists
sizing
snared
theist
morsel
cupful
debars
gibbet
throat
ravine
callup
tamely
holing
clouds
admire
wearer
bagged
cooler
boosts
im bow
mr wax
ah doc
oh ref
as toy
coo ox
tab at
ox yam
my hex
cup by
of bag
ago us
ark my
bag hi
bah as
eat am
dot so
fab to
if not
do off
is oil
dr our
me say

nu spa
ox pun
ah sun
ye rat
ow wet
us wig
re you
nxZxAP
oJwqMO
QfEUFU
qCGzPQ
qAojxE
tUXlog
ElLzKc
kXypEK
mQIuRW
kUDeIA
hWTgjM
zeXBqE
SisnJe
RCeqOA
JdHWMd
NGbUxp
mtQWIL
Tpjblz
jCdLyQ
LpMjYQ
HmVcHN
MelHtk
cswDtu
qATerJ
dzCrSr
RVgOPd
PUWrqk
WmWbAz
gaPHbx
MHCQHE
XnOwlT
OOKPff
owdTDg
aKSsOO
DFXgRK
swHNqS
TkFRPv

eZIFXU
raOBuO
ffWjwD

Group 3:

7ir1s!
stem#7
type9!
4disc$
ewes4$
6mode@
#home0
c@st1@
wean5!
oven3?
?purl2
glim7=
#gave8
reel3$
hers7#
liar1!
pyre2%
step9+
4eyed@
6lien#
fife$9
smug7!
!near0
7lags+
0ruck$
@gage9
$slob1
#yule0
4we@r8
!gals8
swot@5
9@blat
1wail!
chat@6
9weep#
@maxi8
#cult1
?meow6

6this#
4de@l!
@wont9
army0!
!thai7
+tags8
=sell7
8d0se7
&dull3
#dye10
got$49
!dab#7
set@38
45fox!
soy89@
#2m0d£
$sag£6
she%78
0!sap9
8!rat7
lo010!
@#for0
g=HBj6
$jFgR3
yIm+7O
lv?bb5
m@yF4E
4zb?X!
Gr=6Tr
Qg#AA2
DWds6@
3?M?X$
tx@F02
J3AV@w
Our#2G
46=Nyb
aO3&@S
?7Rb35
Sx7#I0
$7P&Fs
2+npx5
kW+2+c
39!5TT
ei#7?&

16JY=+
c9FS7=
5U$Odg
ui63$5
e=fM3V
+4VoUl
FUR1X&
$oK+r3
6JM=H@
EN?3!H
XsI6iC
hbQq9=
C?QWUv
dU$5zN
vv$3wG
&1xClg
P68+QU
7aB?Eh

Group 4:

enem@1!
belts3$
7no#how
vista#0
3soupy%
c@sks72
sands2?
daffy9£
keyed#6
£8flake
speed@5
gaunt8!
b@ste6#
craze2$
7$uncut
fifty5#
heeds=0
shelf+9
tardy#3
8pests!
6porno#
$p@yer8
#manse8

$trial1
sport&5
#obese!
chips#9
%femur0
8trews!
noses0#
valet@7
$prays2
0wo0dy#
petty=8
3husks+
5short%
9swept%
5alloy&
going7&
@upper2
9edged#
gives6#
!rerun1
riffs0£
jiffy8@
@eater0
hussy9#
&crony0
&fasts9
3stead%
$lamas3
4shrug#
parka8&
5gonad£
piton8=
balls+4
=guard9
dr@ws87
sends&1
undue0#
+zAsuK3
nmsU$R6
r?C$&0e
l7rk&PQ
Pqv@7ek
?!n8xTP
8Jhz=G#

kif!N2Q
svP7m@h
ul+b97D
ML5B!l1
AMQ#4cy
ef+!8KX
6icl@3V
7=P$52w
l8Zv!jT
j@by4mk
8ZsmFA#
Ftp?Bm9
ZX39+82
r@4s9s=
Sbd?sZ7
uj8At?b
4or@!nj
pMgkq&1
6k11U$y
k6ekUK&
vB3?g$T
c+xzU1t
2+lDAmx
efQOk#3
HG7ue+s
r?=9CwD
a7wFv
rm5&x!k
KlvXL9=
v1l+zc4
4S8J$ls
cF2ww!O
z2K@pt3

Group 5:

acquire8
opacity5
3arbiter
moha1rs8
0perand8
forfeit4
tourism2
caveats7

9matured
8killers
2oeuvres
externs9
factory3
eroding5
deludes4
lioness9
bulwark0
7watered
m0tored4
morales4
touting9
rippled7
3passing
7infants
9garland
profile4
inhales6
advises1
workout3
r0sette1
rectors8
l1pread3
jutting4
striver8
reining4
respray0
7g1rlish
9spacers
errands0
3beds0re
3starter
prouder5
turning4
flypast8
9finicky
2firmly8
basked10
delude66
7enable7
faroff40
5mender3
77bereft

27inlaws
openly50
sliver00
onuses65
pieced61
60sticky
prince73
cavern16
aGMtbCH2
F6kqfJzO
yKclk6k5
6KLDePAU
D2GgJniA
7YdbUQAB
ahuBUVY6
mlKsy1nJ
J4Ke4M4f
l0Gw5RSt
JJhrWZ7B
rvc6qXf2
4Fl2i1jA
7yaJwhoY
nDt8tgXi
RdkEdC1z
X94dgoOm
I7gLoIgs
cK1iyjt2
jp7IoAmh
2B2GkApR
coC26bP6
1ClEtOVa
cS2TjK9U
js2vhxp6
wJkt3aWJ
lfUXH6sV
v4520n4B
OlC1SBbv
TEJ7donY
eLwgI5EA
xLEvk2ga
VHtZ0pHz
ICWwRf7Y
zl2m7LRc
eUvL1oYT

i1JmqqnA
8NJxW8Yx
NzO1EPTH
ZAAXe3lS

APPENDIX C – HASHING PROGRAM

```
import bcrypt
import hashlib

with open("Basic.txt", "r") as P, open("MD5.txt", "a") as H, open("Bcrypt.txt", "a") as B:

    P1= P.readlines()
    for x in P1:

        Pass = x.rstrip()
        #print(Pass)

        #MD5 HASHING
        res = bytes(Pass, 'utf-8')
        md5_has = hashlib.md5()
        md5_has.update(res)
        H.write(Pass + ": ")
        H.write(md5_has.hexdigest())
        H.write('\n')


        salt = bcrypt.gensalt()
        hashed = bcrypt.hashpw(res, salt)
        B.write(Pass + ": ")
        B.write(" Salt : ")
        B.write(salt.decode("utf-8"))
        B.write(" Hash : ")
        B.write(hashed.decode("utf-8"))
        B.write('\n')

P.close()
H.close()
B.close()
```

APPENDIX D – RAINBOW CREATION BATCH FILE

RainbowCreation.bat

```
rtgen md5 group-1 1 8 5 2400 24652134 0
rtgen md5 group-1 1 8 4 2400 24652134 0
rtgen md5 group-1 1 8 3 2400 24652134 0
rtgen md5 group-1 1 8 2 2400 24652134 0
rtgen md5 group-1 1 8 1 2400 24652134 0
rtgen md5 group-1 1 8 0 2400 24652134 0
rtgen md5 group-2 1 6 5 2400 24652134 0
rtgen md5 group-2 1 6 4 2400 24652134 0
rtgen md5 group-2 1 6 3 2400 24652134 0
rtgen md5 group-2 1 6 2 2400 24652134 0
rtgen md5 group-2 1 6 1 2400 24652134 0
rtgen md5 group-2 1 6 0 2400 24652134 0
rtgen md5 group-3 1 6 5 2400 24652134 0
rtgen md5 group-3 1 6 4 2400 24652134 0
rtgen md5 group-3 1 6 3 2400 24652134 0
rtgen md5 group-3 1 6 2 2400 24652134 0
rtgen md5 group-3 1 6 1 2400 24652134 0
rtgen md5 group-3 1 6 0 2400 24652134 0
rtgen md5 group-4 1 7 5 2400 24652134 0
rtgen md5 group-4 1 7 4 2400 24652134 0
rtgen md5 group-4 1 7 3 2400 24652134 0
rtgen md5 group-4 1 7 2 2400 24652134 0
rtgen md5 group-4 1 7 1 2400 24652134 0
rtgen md5 group-4 1 7 0 2400 24652134 0
rtgen md5 group-5 1 8 5 2400 24652134 0
rtgen md5 group-5 1 8 4 2400 24652134 0
rtgen md5 group-5 1 8 3 2400 24652134 0
rtgen md5 group-5 1 8 2 2400 24652134 0
rtgen md5 group-5 1 8 1 2400 24652134 0
rtgen md5 group-5 1 8 0 2400 24652134 0
```

APPENDIX E – COMPLETE RESULTS

	Group 1	Group 2	Group 3	Group 4	Group 5
Web-based	30%	33%	6%	1%	40%
JTR rockyou.txt	29%	22%	0%	0%	24%
JTR md5-decrypt-awesome	30%	33%	0%	0%	39%
RainbowCrack	1%	100%	42%	1%	1%
Combined maximum cracking potential	30%	100%	44%	2%	44%