Abstract:

With the advent of digital-age passwords have become the most prevalent authentication scheme. This is because of their superior versatility and deployability in comparison to other authentication mechanisms. Unfortunately, humans still struggle with creating strong passwords that are per the guidelines listed in standard password policies. Users, in general, are reluctant to follow these policies and thus end up creating passwords that contain predictable patterns or reuse the same password across multiple platforms which in turn significantly reduces the work of hackers. This behaviour establishes the need for password analysis tools that users can utilize to determine the strength of their passwords. This project report explores the use of machine learning algorithms to reduce the computation needs of traditional rule-based password analysis tools and to make them more flexible.

Contents:

1. Introduction:

In today's world, with the ever-increasing computation power of processors, utilization of graphical processing units for password cracking, and development of efficient and effective password cracking algorithms, the secureness of password policies which were initially designed to be resistant to simple brute-force attacks is questionable. It is a fact that many tech giants with sufficient resources have dealt with this issue by using advanced slow hashing algorithms such as bcrypt, PBKDF2 e.t.c to keep their password database secure. However, this does not solve the issue at hand as smaller firms and startups are usually the ones at the receiving end of such password cracking hacks. These companies due to lack of sufficient resources still use common MD5-based hashing algorithms(e.g SHA1). Therefore, there is a need for the development of password analysis tools that can quantify the strength of passwords. People can use these tools to make their passwords secure in case the company's password database is hacked and then leaked by the hacker.

Many rule-based password strength classifying tools have already been developed and are available on the internet. These tools check for various patterns in a given password to determine its strength(e.g What is the length of password?, What is the key-space of password?, Does the password contain any dictionary word? e.t.c). However, with each advance in password cracking algorithms used by the hackers, new rules need to be added to these rule-based password analyzers to ensure the measurement of password strength given by them is valid. This repeated addition of pattern testing rules has already made modern rule-based password analysis tools computationally intensive and will continue to do so as long as the passwords keep on getting more and more complex which is certain. One way to solve this issue is to introduce Machine Learning components into these password analysis tools. These Artificial Intelligence components will cut the computational needs of these tools and prevent them from further increasing provided they are trained with the latest passwords.

This project aims to develop a hybrid tool based on this philosophy which combines the flexibility of Machine Learning algorithms with traditional rule-based pattern testing algorithms to create a password analyzer that is capable of improving itself provided it is trained with the latest passwords.

2. Background and Motivation:

In recent years many Machine Learning based password strength analysis tools have been developed. Most of these software tools use various classification algorithms such as Logistic regression[1], Ensemble Learning(Decision tree learning) e.t.c to create models that classify a given password by assigning a specific label value to it based on its complexity. The class prediction models are created by training these classification algorithms using word-lists of real user passwords. These password lists are leaked over the internet by hackers after successfully hacking into various company databases. E.g's of these password lists include rockyou(2009 breach of online games service RockYou, 14 million leaked passwords), hashkiller, 000webhost e.t.c.
These password classifiers based purely on Machine Learning classification algorithms have been very successful(> 75% success rate) in classifying passwords. However, there is an inherent issue with these models. They can only assign a strength class to a given password i.e they can only tell us whether our password is strong/weak or strong/satisfactory/weak. These models cannot give us the measurement of the strength of a password in terms of real numbers or integers. To deal with this issue many attempts at regression-based prediction models have been made but due to the amount of randomness, a password can exhibit these models are prone to overfitting and hence give bad prediction results.
In addition to these classification models, some attempts have been made to develop password strength classifiers based on clustering algorithms(Kernal Density Estimation(KDE) [7]). This model explores the clustering of passwords in the space whose dimensions are defined by structural features extracted from each password, which are transformed and down-selected using Principal Component Analysis (PCA).

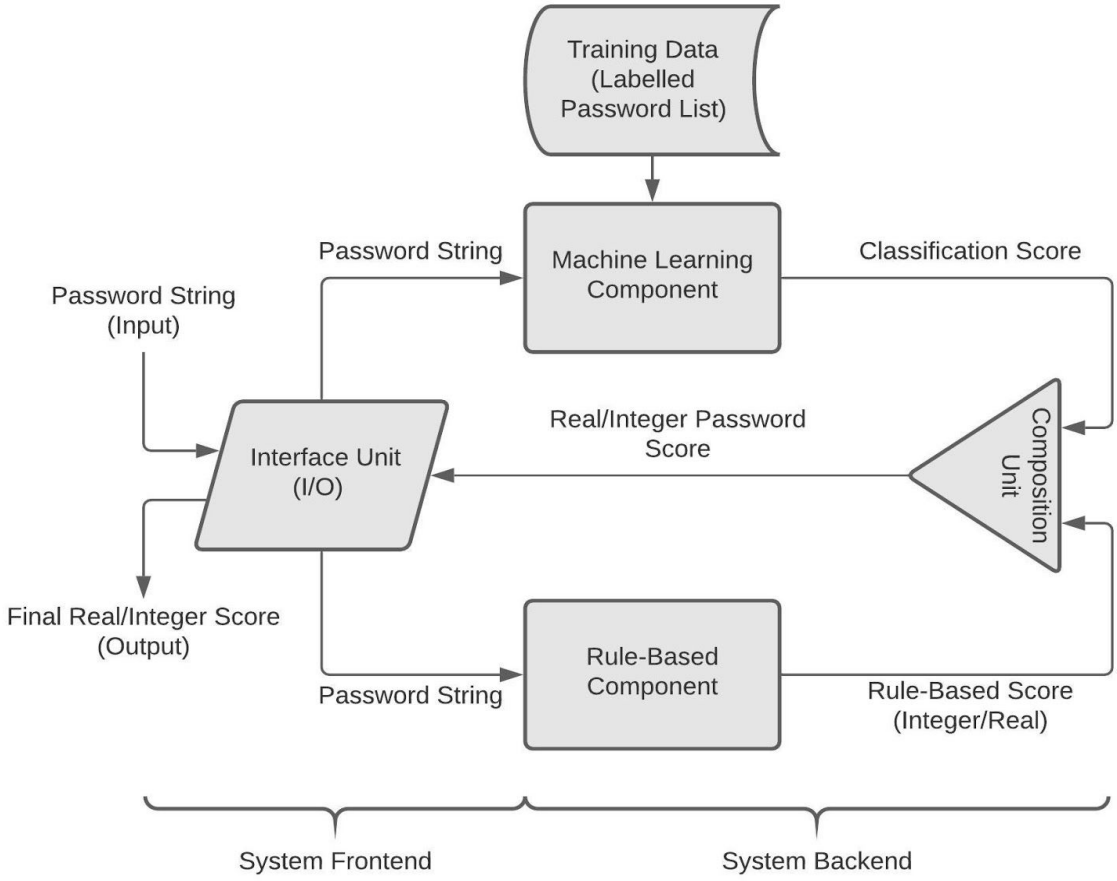
Considering the above-mentioned issues, it is clear the problem statement is to effectively inculcate the classification algorithms into our analysis tool without compromising the success rate of prediction results.

3. Approach:

This project report investigates the possibility of leveraging machine learning classification algorithms in the future to cut short most computation-intensive rule-based pattern testing algorithms in traditional password strength analyzers. This hybrid amalgam will at least comprise of the following four components:

- A Machine Learning component utilizing classification algorithms.
- A light-weight Rule-based component using specific pattern testing algorithms to assign a real-value/integer score to a password.
- A Composition/Integration unit that combines the scores of the aforementioned two components using a specific strategy to generate the final score allocated to the password.
- An Interface unit for password input and score output.

3.1 Block-Diagram:



3.2 Machine Learning Component:

The machine learning component essentially comprises class prediction models created by training various classification algorithms using password lists leaked over the internet. Some of these classification algorithms are as follows:

- Logistic Regression
- K-Nearest Neighbour(KNN) classification
- Support Vector Machine(SVM)
- Naive Bayes
- Decision Tree classification
- Ensemble Learning(Random Forest classification)

These classification algorithms use different strategies to classify passwords with each strategy being based on or more classification parameters. Some of these parameters are Cross-Entropy, Cosine Similarity, Information Gain, Gini Index, Gini Ratio, Chi-Square values e.t.c.

3.2.1 Data-set Description:

In this project, as has been already mentioned the data-sets we are using are leaked password lists. However, there is a big issue with these lists that have not been addressed. These lists comprise of millions of passwords but these passwords have not been classified. In other words, our current data-set comprises of passwords without any strength label/class assigned to them.

Now the question remains, What medium should we use to classify these passwords in our data-set such that the results of classification are acceptable by current standards. After all, if the passwords in the training data-set have not been properly labelled then the prediction models created from these data-sets will also give bad results. Here we have used a tool called PARS(Password Analysis and Research System)[2][3] developed at Georgia Tech University. This tool has most of the commercial **Rule-based** password strength classifiers integrated into it. After feeding it the password list we got separate classification results of each of the classifiers integrated into this tool. Each of the classifiers sorted the passwords into three classes(0-Weak, 1-Average, 2-Strong). We then took the intersection of these classification results to obtain our final data-set. In other words, we only took those passwords for which all classifiers were in agreement (i.e passwords which got the same label from all classifiers). An example of the resulting data-set is as shown in fig(i).

password	strength
kzde5577	1
kino3434	1
visi7k1yr	1
megzy123	1
lamborghin1	1
AVYq1lDE4MgAZfNt	2
u6c8vhow	1
v1118714	1
universe2908	1
as326159	1
asv5o9yu	1
612035180tok	1
jytifok873	1
WUt9IZzE0OQ7PkNE	2
jerusalem393	1
g067057895	1
52558000aaa	1
idofo673	1
6975038lp	1
sbl571017	1
elyass15@ajilent-ci	2
intel1	0
klara-tershina3H	2
czuodhj972	1
faranumar91	1
cigicigi123	1

fig(i)

3.2.2 Feature Matrix Modification:

As of now, our data-set looks as shown in fig(i). The feature matrix contains 1 column(passwords) and a dependent vector(strength) which classifies passwords in 3 labels(0-Weak, 1-Average, 2-Strong). Now if we were to use this data-set to train our classification model, we would get bad prediction results. This is because our only column in the feature matrix(passwords) has a huge domain. This will continue to be true even if we restrict the length of passwords to a specific range because of the permutation of characters in the string.

Now to deal with this issue, instead of taking the entire string of password as a token, we take individual characters inside the password string as tokens[1]. In this way, we

have a feature matrix containing multiple columns with each column containing only a single character. Thereby reducing its domain of values. After carrying out this modification we have our final data-set.

3.2.3 Technologies used for implementation:

- Python Libraries:
    1. Sci-kit-Learn
    2. Numpy
    3. Pandas

- Mysql for manipulating database containing training data-set

3.3 Rule-Based Component:

The rule-based component is a light-weight unit that contains a collection of tests that have been designed and manually programmed by the developer. Each of these tests check a given password based on certain criteria. Some of these criteria are listed as follows:

- What is the length of the password?
- Does the password contain letters/numbers only?
- How many repeated characters are there in the password?
- How many consecutive uppercase/lowercase letters are there in the password?
- How many consecutive numbers/special characters are there in the password?
- In addition to these common pattern tests, there are specific tests that check whether the given password is based on a specific mask.

The password string is evaluated by these tests and then assigned a real/integer score based on the number and type of tests in which it gets the desired results. The type of tests plays a role because not all tests have the same contribution towards the overall score of the password i.e different tests have different weightage.

There is a reason for having this light-weight rule-based component in addition to the machine learning model. The machine learning model assigns a specific class to a password based on its complexity. However, there are password strings such as “momokids@15”, “momof3g8kids” e.t.c that might seem complex based on string length, key-space e.t.c but these passwords can be easily cracked using Mask Attacks[4], Rule-based Attacks[5], Combinator Attacks[6] e.t.c. The machine learning model will sort these passwords in Average or Strong classes which will certainly be a wrong prediction. Therefore we need this rule-based component to perform pattern-based tests aimed to detect these types of passwords.

3.3.1 Technologies used for implementation:

- Python Libraries:
    1. Numpy

3.4 Composition/Integration Unit:

The Composition unit takes the classification score of a given password allocated by the machine learning unit and the integer/real score of the rule-based unit as its inputs. It combines these scores using a specific strategy to generate the final real/integer score assigned to the given password.

3.4.1 Technologies used for implementation:

- Python Libraries:
    1. Numpy

3.5 Interface Unit:

The function of the interface unit is to take password inputs from users and output final scores assigned to the password. In this project we are using a web application interface i.e we are hosting our analysis tool on a web server. The user will have access to this tool via a website URL. The users will input the password strings in the appropriate section of the web-page which will be captured and then forwarded to the back-end of the website where the running password analysis tool will assign a score to this string. This score will then be transmitted to the website front-end and will be displayed on the web-page.

3.5.1 Technologies used for implementation:

- Website Backend:
    1. Django framework

- Website Frontend:
    1. HTML
    2. CSS and CSS framework (Bootstrap)
    3. Javascript

Project-Timeline Group-9

	October	November	December	January	February	March	April	May	June
Project Research & Planning									
Machine Learning Component Development									
Rule-Based Component Development									
Composition Unit Development & Testing									
Interface Unit Develpment & Unit Testing									
Interface Integration									
Overall Project Testing & Debugging									

Legend: JAMYANG | AQUIF | COLLABORATIVE WORK

7. References:

[1]	Ahmed, Faizan (2016). Machine Learning based Password Strength Classification. Available from World Wide Web: https://medium.com/@faizann20/machine-learning-based-password-strength-classification-7b2a3c84b1a6
[2]	E.g of PARS processed dataset. Available from World Wide Web: https://www.kaggle.com/bhavikbb/password-strength-classifier-dataset
[3]	Ji, S., Yang, S., Wang, T., Liu, C., Lee, W., & Beyah, R. (2015). PARS: A Uniform and Open-source Password Analysis and Research System. ACSAC 2015. Available from World Wide Web: https://www.semanticscholar.org/paper/PARS%3A-A-Uniform-and-Open-source-Password-Analysis-Ji-Yang/99b9bff925fdab9c4680f5667e7e6bc93cde2980
[4]	Mackay, W. (2016). How to Perform a Mask Attack Using hashcat. Available from World Wide Web: https://www.4armed.com/blog/perform-mask-attack-hashcat/
[5]	Mackay, W. (2016). How to Perform a Rule-Based Attack Using hashcat. Available from World Wide Web: https://www.4armed.com/blog/hashcat-rule-based-attack/
[6]	Mackay, W. (2016). How to Perform a Combinator Attack Using hashcat. Available from World Wide Web: https://www.4armed.com/blog/hashcat-combinator-attack/
[7]	Todd, M. (2016). An Investigation of Machine Learning for Password Evaluation. Available from World Wide Web: https://www.semanticscholar.org/paper/An-Investigation-of-Machine-Learning-for-Password-Todd/d40dee322446d8a3e45a622e1cf80f990976627c