Introducing a Machine Learning Password Metric Based on EFKM Clustering Algorithm

Omar Saad Almousa[1] and Hazem Migdady[2]

[1] Computer Science Department, Jordan University of Science and Technology, Jordan
[2] Computer Science and Management Information Systems Department, Oman College of Management and Technology, Oman

***Abstract*:** we introduce a password strength metric using the Enhanced Fuzzy K-Means clustering algorithm (EFKM henceforth). We train the EFKM on the OWASP list of 10002 weak passwords. After that, we maximize the optimized centroids to develop a password strength metric. We validate the resulting meter by contrasting with three entropy-based metrics using two datasets: the training dataset (OWASP) and a dataset we collected from the GitHub website that contains 5189451 leaked passwords. Our metric can recognize all the passwords from the OWASP as weak passwords only. Regarding the leaked passwords, the metric recognizes almost the entire set as weak passwords. We found that the results of the EFKM-based metric and the entropy-based meters are consistent. Hence the EFKM metric shows its validity as an efficient password strength checker.

***Keywords*:** Password Strength Metric, Clustering, EFKM, Entropy, OWASP, Weak Passwords, Leaked Passwords**.**

1. Introduction

Passwords provide a mechanism to verify authenticity [1]. They are vital for any system, whether it is online or offline [2]. Using passwords is the most common technique for user authentication for three reasons: deployability, usability, and security [3]. The authors of [4] [5] believe that passwords will remain one of the top user authentication techniques during the foreseeable future, despite the major shortcoming of this approach, which is the difficulty of creating passwords that are strong and easy to remember, Moreover, they argue that no alternative approach can keep the advantages of passwords techniques introducing no problems.

Having strong enough passwords is a very important factor for an account to be safe and well-protected. Identifying the term "Strong Enough Password" is something not clear nor settled, and this explains the existence of a wide range of policies that control creating and changing passwords. Nowadays, systems provide policies that guide passwords` creating process. Such policies consider the crucial attributes and require each user to satisfy them properly. So, it is possible to say that the user is not completely free to choose and design the password [6].

Even if a system applies a mandatory and tough policy, the threat of creating weak passwords is still possible because cyber threats do not relate only to the quality of passwords, they also relate to other factors connected to the behavior of the user, such as using personal information passwords, passwords lending, reusing or writing them down. In [9], the authors introduced a statistical study that reveals that 53% of the users use the same set of passwords through different systems, 21% lend their passwords, and 17% wrote the passwords to remember them when needed.

The authors of [7] suggested a criterion to evaluate the quality of the password: "how hard it is for a third part to predict a password". They found that the correlation between the easiness of password estimation and its strength is a positive correlation. However, a strong password (i.e., good password) is that one that satisfies the criteria of the policy under consideration. In any system, having no password policy at all or having an optional one would increase the threat of creating weak passwords, and this because of several reasons: the laziness of users, lack of education, lack of security awareness… etc. [8].

One way to force users to choose strong passwords is the use of password meters. The primary goal for a password meter is to direct the users to satisfy all the instructions and the rules of the policy under consideration while creating or updating passwords. So, having clear and specific passwords policy is very important for the meter to work properly. The computer systems used various policies, and most of them focus on the components of the password, such as letters, digits, special symbols, the length of the password itself, and character frequency. Other policies focus on the user behavior that directly affects the type and meaning of a password. However, the former policies are easier to implement and deploy in the systems since they require conventional skills for proper implementation. Whereas the later policies need revolutionary techniques and skills related to machine learning and data mining methods to satisfy them.

In this paper, we introduce a machine learning-based password meter. We applied the EFKM algorithm to fit the underlying structure of the OWASP dataset of 10002 weak passwords. EFKM algorithm generates a set of centroids that each one of them represents a cluster of weak passwords. We optimized the result of EFKM into one unified centroid by maximizing the resulting centroids. After that, we used that unified centroid as the basis to develop our EFKM-based meter. The EFKM-meter recognizes unobserved passwords and evaluates them as strong or weak passwords.

In the next section, we present the most significant related work. In Section 3, we present our methodology to develop the proposed EFKM password strength meter. Then we give the evaluation methodology in Section 4. In Section 5 we show the experiments we carried out and the results we found. We discuss the validity of the EFKM meter in section 6. Finally, we conclude the paper in section 7.

2. Related Work

Most of the systems require users to create diverse and complex passwords which make them difficult for users to memorize and retrieve [3][10][11][12] especially nowadays

that witness an overwhelmed growth of accounts for each user [12] and this explains the mentioned habits of the users when creating passwords. As part of the efforts to interpret how users choose their own passwords, an approach that applies the password graphs. Such approaches provide important properties of passwords and introduce beneficial clusters of them.

In [10], the authors discussed the effect of users' behavior on choosing passwords and its strength. Several researches and studies attempt to understand the user-chosen passwords. For instance, [13] stated the major reasons that cause the problems related to the text-based authentication. They believe that creating a memorable password is a challenging matter. Hence, they suggested an approach based on the psychological factors of the user to encourage creating strong enough and memorable passwords.

However, [14] argue that the crucial reason in the issues of secure passwords is reusing the same password for multiple accounts. Thus, guessing a password from one account implies that all the other accounts are accessible with no efforts. Furthermore, the authors of [15] found that almost 60% of all users use the same password through multiple systems, whereas [16] noticed that most users are not aware of the best practices in creating strong passwords.

A fruitful study [17] presented an analysis of the habits of the users over Chinese network. The study measured the strength of passwords through a comprehensive analysis of their length, type, and "other variables", and hence, the existence of some repeated patterns in the analyzed passwords. Thus, the need of a meter to evaluate the password security is a crucial demand [18]. A password meter (i.e. password strength checker) is as a common way to overcome such threats. Passwords meters provide valuable feedback for users on when creating or changing passwords to improve the password strength. Many meters are available with different approaches and policies. There is a necessity to study the effect of the meter-based passwords regarding security and usability according to [19]. They found that the users place high importance on satisfying the policy of stringent meters with visual feedback bars. Such findings help researchers to develop efficient meters that overcome the major problems towards strong passwords. The authors of [20] also mentioned that no meter is absolutely better than any other meter among those they studied. Moreover, they stated that each meter has its own strength and weakness features. So, they extracted a meter that uses and combines the strength features from all other meters. The authors of [21] developed an efficient meter, they used data-based methods to interpret how users create their own passwords and how hackers can predict them. They believe that a valuable meter is that one which helps users to choose a strong and memorable password by balancing the importance of those two factors. And in the same context, [5] suggested a metric to handle the problem of containing personal information in passwords. They applied a segmentation algorithm to avoid any personal attribute of the user that highly correlates to the selected password. The researchers in [22] categorized the meters into two categories: 1- Industrial meters and 2- Academic meters. They believe that even though the industrial meters are not accurate enough, since they use a very simple heuristics to help users in creating the needful passwords, they are better than the academic meters that are still far from any desired satisfaction. The authors noticed that most of the users recall passwords similar or identical to what they used before. So, to overcome the shortcomings of the existing passwords meters, they introduced a password meter and made the needed models of the users' behavior by training phases using a set of leaked passwords. For any meter, accuracy is as a very important feature to be in passwords evaluation since the current meters are not reliable enough. Till now it is not clear how to evaluate the accuracy of a password strength meter. Hence, a group of attributes should be satisfied to develop an accurate meter. Hence, [23] emphasized that the lower the accuracy of a meter, the higher the risk of passwords leakage.

A common approach to evaluate a password in terms of its strength level is passwords rating. In [24] the authors categorized the techniques of the password strength rating into two major categories: 1- machine ratings and 2- human ratings. After that, they implemented a survey to measure how such ratings affect users' trust. They found that users trust machine ratings more than human ratings with a level of subjectivity. The principal reason for password leakage according to [25] is because of the strength of the password. Thus, weak passwords are subject to get leaked more than strong passwords. The authors of [6] suggested a meter measures the strength of a password in two ways, namely: password entropy and password quality. After that, they concluded with two rules: high entropy passwords are also high-quality passwords, and low-quality passwords are low entropy passwords.

3. Development Methodology

The methodology of developing our password meter has three phases: 1- dataset preprocessing 2- training of the learning machine 3- developing the password meter.

1- Datasets and Preprocessing:

In order to introduce a reliable enough password meter, we believe that using machine learning and data mining techniques is a very useful and efficient approach to develop a dynamic metric that can handle any suggested password and rate its quality effectively. Such techniques and methods require convenient data sets to have the machine well trained and prepared for any possible unobserved instances. So, we will use two datasets: 1- OWASP dataset that we used to develop the EFKM metric and 2- Github dataset that we used to evaluate the metric. For presentation, we start with presenting the later one and then continue by describing the training dataset in order to maintain a better flow of ideas.

- *Github Leaked Passwords:* we later use this dataset for meter validation purposes. It contains 5189451 leaked passwords from several websites and obtained from GitHub website. As expected, we classify the characters that compose any password into four categories, namely: Lowercase, Uppercase, Digits, and Symbols. Figure 1 shows the probabilities of character categories in this dataset.

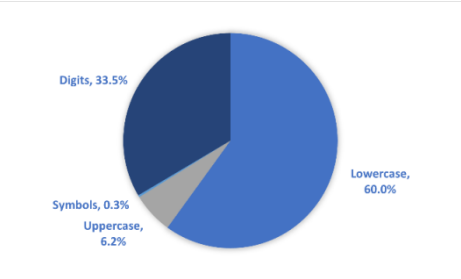


Figure 1. Probabilities of Characters Categories in GitHub Dataset

- *OWASP Weak Passwords [26]:* we use this dataset to train the EFKM algorithm in order to develop the suggested metric. The dataset contains 10002 passwords, all of them are weak passwords in the form of plain text (i.e. text-based values). Thus, to prepare it for further usage we will map it to represent it as numerical vectors, each one has five discrete features: Length L∈3, 4, 5, …, 16 that tells how many letters, digits and symbols a password contains. Lowercase LC∈{0, 1, 2, …, 12} and Uppercase UC∈{0, 1, 2, …, 8} respectively represent the number of lowercase and uppercase letters in each password. Digits D∈0, 1, 2… 8, Symbols S∈ {0, 1, 2, …, 12}. Figure 2 below illustrates the distribution of each features' values in the OWASP dataset.
  - Figure 2 illustrates that the portion of the uppercase letters regarding all characters we use in the passwords dataset is about %0.4, whilst it is %0.1 for the symbols. Such distribution makes sense regarding the weakness level of these passwords.
  - However, having the training dataset with this size and presented in this manner produces a complicated underlying structure that cannot easily be recognized or understood. Hence, applying a machine learning technique for data mining application is a convenient option to use and understand the dataset properly.

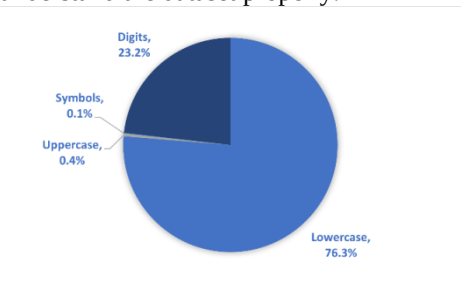


Figure 2. Probabilities of Characters Categories in OWASP Dataset

2- Training of the Learning Machine:

As mentioned earlier, the dataset we have here contains passwords that are weak passwords only that do not have the same level of weakness. Some are extremely weak, while some are very weak or weak. As any similar dataset, the instances in the OWASP set altogether form some patterns and underlying structures that are difficult to catch due to their nature and size. In this manner, we can use this structure and its intercorrelations to introduce a meter that evaluates the strength level of unobserved passwords according to its training on the OWASP dataset.

Since all the instances are weak passwords, we use a clustering technique to identify groups of similar instances. To that end, we used the Enhanced Fuzzy K-Means (EFKM) that the authors of [27] introduced. The EFKM technique involves a modification on the traditional FKM. EFKM improved the performance of the clustering and the calculations of the membership values.

The training session of the EFKM aims to produce n clusters, where n=3, 4, 5, 6. A tuple of five centroids (i.e. quintuple henceforth) represents each cluster. Recall that we describe each of the passwords by a vector of five features. Therefore, each of the runs of EFKM generates a set of n quintuples denoted by C_n. We denote the i^{th} quintuple in the nth cluster by $C_{in}=(C_{in1},C_{in2},C_{in3},C_{in4},C_{in5})$. For instance, the set of centroids that EFKM generates for n clusters is $C_n=\{C_{1n},C_{2n},\ldots,C_{nn}\}$, such that C_{2n} is the second centroid out of n centroids. Moreover, the second quintuple of the EFKM run to generate 5 clusters that we denote by $C_{25}=(C_{251},C_{252},C_{253},C_{254},C_{255})$, hence C_{253} represents the third value of the second centroid produced by the run of the EFKM that generates five centroids (clusters).

In the suggested framework, the training phase starts by feeding the passwords from the training dataset (OWASP) into the EFKM. After that, EFKM generates the clusters by optimizing a set of quintuple centroids, such that a single quintuple centroid is produced for each cluster. To analyze and understand the entire process to develop the proposed meter, we implemented four training sessions for the EFKM under different parameters. In each training session we changed only one parameter, which is the number of clusters n. Hence, we train the EFKM four times, starting with 3 clusters up to 6 clusters in the last training session.

3- EFKM Meter Development:

We develop an efficient enough meter via a combination of the quintuples we had from a specific training session. So, we defined an extra centroid (C_{xn}) for each n. C_{xn} is a quintuple calculated by maximizing the set of the optimized centroids that we generate by EFKM trainings for a given n. More precisely, C_{xn} is:

$$C_{xn}=(\max(C_{1n1},C_{2n1},\ldots,C_{nn1}),$$
$$\max(C_{1n2},C_{2n2},\ldots,C_{nn2}),$$
$$\max(C_{1n3},C_{2n3},\ldots,C_{nn3}),$$
$$\max(C_{1n4},C_{2n4},\ldots,C_{nn4}),$$
$$\max(C_{1n5},C_{2n5},\ldots,C_{nn5}))$$

Where max(.,.,.,.,.) finds the maximum value of its five parameters. For instance, the set of centroids generated by EFKM for 3 clusters (C_3) as illustrated in Table 1.

Table 1. Three Clusters Quintuples

Quintuple 1	C_{131}	C_{132}	C_{133}	C_{134}	C_{135}
Quintuple 2	C_{231}	C_{232}	C_{233}	C_{234}	C_{235}
Quintuple 3	C_{331}	C_{332}	C_{333}	C_{334}	C_{335}

Table 1 is another way to show the set of the three quintuples produced by EFKM for three clusters. C134 denotes the fourth term of the first centroid out of three clusters. The actual values (rounded up to 3 decimal places) that we obtained by running EFKM for three clusters are in the following table:

C_{13}	7.416	0.065	0.009	0.004	7.337
C_{23}	5.565	5.423	0.018	0.002	0.121
C_{33}	7.646	7.539	0.012	0.001	0.094

The last step is to find C_{x3} by maximizing C_3. There for the values of C_{x3} are as follows:

C_{x3}	7.646	7.539	0.018	0.004	7.337

Hence, after applying the maximization step over all sets of quintuples from the training session, four maximized centroids will be produced:

C_{x3}	7.646	7.539	0.018	0.004	7.337
C_{x4}	7.992	7.550	0.073	0.012	7.946
C_{x5}	8.186	8.085	0.030	0.006	8.012
C_{x6}	8.047	7.663	0.070	0.011	8.024

The goal of this maximization step is to harden the optimized centroids as a password meter. Such an approach makes the proposed meter more reliable since it will not be easy to be satisfied (i.e. any accepted password satisfies tough criteria).

To evaluate a password s using EFKM of n clusters, we first calculate the five features (f_1, f_2, f_3, f_4 and f_5) of the password s getting a quintuple. We denote the result with $ff(s)=(f_1,f_2,f_3,f_4,f_5)$. Then we calculate the distance between the password's quintuple ff(s) and C_{xn}. To calculate the distance between $ff(s)=(f_1,f_2,f_3,f_4,f_5)$ and $C_{xn}=(C_{xn1}, C_{xn2}, C_{xn3}, C_{xn4}, C_{xn5})$ we use the following function:

$$distance(ff(s),C_{xn})=(f_1-C_{xn1},f_2-C_{xn2}, f_3-C_{xn3},f_4-C_{xn4},f_5-C_{xn5})$$

$$= (d_1, d_2, d_3, d_4, d_5) \quad (1)$$

Where d_i is just a notation for the result to use it later. Note that this function produces signed values (positive and negative) such that the lower the value, the weaker the password, and vice versa. Finally, we map *distance(.)* to a single value. This mapping enables us to perform systematic comparisons between EFKM metric and the entropy-based metrics that we show later. We base this mapping on the complements of probabilities of each of the features in the dataset of five-million leaked passwords shown in Figure 1. The reason behind the complementation step is to give more value to unlikely characters. We derived this from Shannon's uncertainty principle. The final EFKM metric for a password s is therefore:

$$\text{EFKM-Metric}(s) = p^c(\text{lower}) \times d_2 + p^c(\text{upper}) \times d_3 + p^c(\text{digits}) \times d_4 + p^c(\text{symbols}) \times d_5$$

Where $p^c(.)=1-p(.)$, is the probability complement, and the d_is are as denoted in (1) above. Note that we omit the first feature from our mapping, recall that the first feature represents the length of the password, and the length of passwords have no probability in our model.

4. Evaluation Methodology

To evaluate the EFKM meter, we use the following three entropy-based metrics:

1- NIST entropy: this metric is the well-known NIST entropy [28] that is an approximation to Shannon's entropy. For any password, we calculate its NIST as follows:

a) 4 bits for the 1st character.
b) 2 bits for the next 7 characters.
c) 1.5 bits for the next 12 characters.
d) 1 bit for the remaining characters.
e) 6 bonus bits if the password contains a combination of an uppercase and a digit or a symbol.
f) 6 bonus bits if the password passes a dictionary check (we have not considered this step in our calculations).

Figure 3 shows the histogram for calculating NIST entropy for leaked passwords. It shows that most of the leaked passwords are weak.

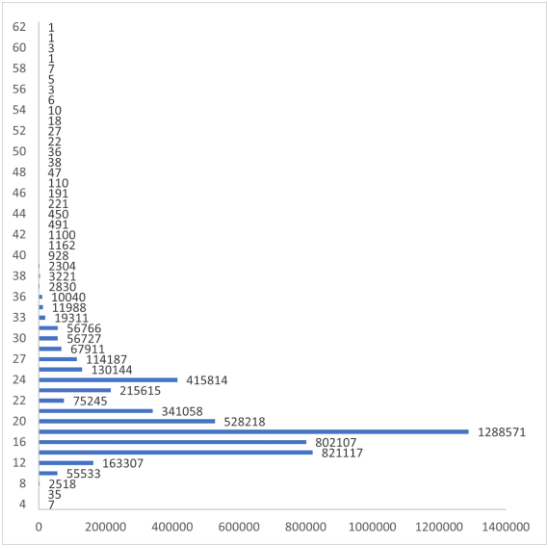


Figure 3. NIST Metric for Leaked Passwords

2- Uniform entropy: this metric is based on the assumption that all passwords' characters have equal probabilities, i.e., have a uniform distribution and therefore have equal probabilities. This is basically Hartley's entropy with base 2. More precisely, for a password *s* of length *n*, the uniform entropy of *s* is: $Hu(s) = log_2 n\ H_u(s) = log_2 n$

Figure 4 shows the histogram for calculating Uniform entropy for leaked passwords. It again shows that most of the leaked passwords are weak.

3- Pure entropy: this metric is a trial to implement Shannon's entropy. We calculate it based on the probabilities of character category in the set of the leaked passwords. The probabilities are given in Figure 1 and Shannon's entropy for a password $p=s_1s_2...s_n$ is: $H(p) = -\sum_{i=1}^{n} p(s_i)log_2p(s_i)$

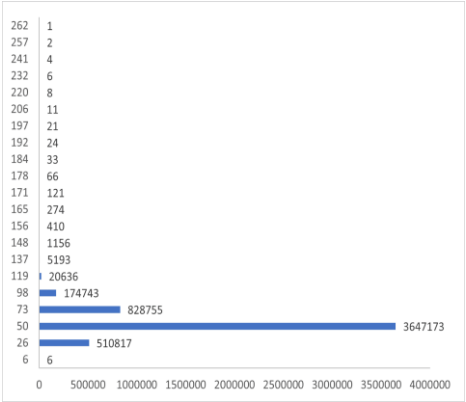


Figure 4. Uniform Metric for Leaked Passwords

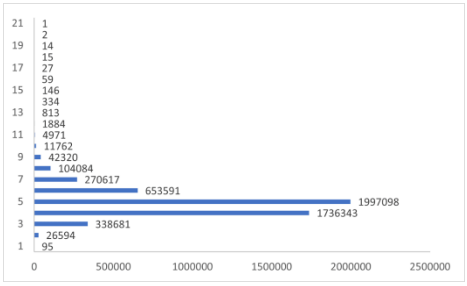


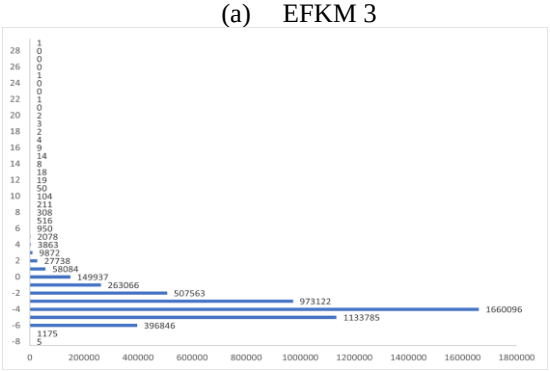
Figure 5. Pure Metric for Leaked Passwords

Figure 5 shows the histogram for calculating Pure entropy for leaked passwords. It also shows that most of the leaked passwords are weak.

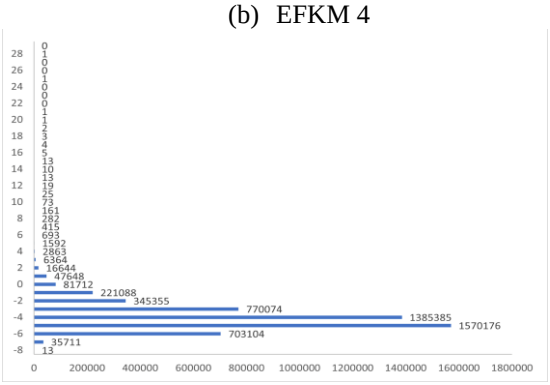
5. Experiments and Results

Up to this point, we explained how to compute the EFKM meter. We also presented the evaluation methodology of the EFKM meters that we use based on three entropy-based metrics for passwords strength. As explained earlier, we use any of the maximized quintuples that we have from EFKM meter development phase with any unobserved password in order to identify how far it is from that quintuple. We do so to state the strength or weakness of the unobserved password. Now, we present our experimental results that we obtained from applying the EFKM Metric on the set of leaked passwords. More precisely, we applied four different versions of the EFKM metric. We obtained those four versions from running EFKM algorithm for 3, 4, 5 and 6 clusters. Figure 6 illustrates the results after calculating the distances between each password in the leaked passwords dataset and the EFKM meter versions. More specifically, the figure shows the distribution of the passwords by representing the occurrences of the passwords regarding each distance we have with the maximized quintuples generated by EFKM for 3, 4, 5 and 6 clusters (sub-figures a, b, c, d respectively). The y-axis in each figure represents the distance between a password and the maximized quintuple, such that the negative distance means the password is weak (i.e. the lower the distance, the weaker the password and vice versa). It is possible to make the results from figure 6 clearer by categorizing the leaked passwords into six categories, such that each password is: Extremely Weak, Very Weak, Weak, Strong, Very Strong, or Extremely Strong. The four sub-figures that we have in Figure 7 represent the results of leaked passwords categorization, such that figure 7-a, 7-b, 7-c and 7-d illustrate the results of categorization after applying the maximized centroids that we produced from the training sessions under 3, 4, 5, and 6 clusters respectively. Figures 7-a, 7-b and 7-c show that increasing the number of clusters increases the number of extremely weak passwords. However, figure 7-d contradicts this conclusion. We have this situation for two reasons: 1) the optimal number of clusters we need. For any clustering problem, it is difficult to specify the most suitable number of clusters. Specially that one should determine in advance as an input which makes it one of the most significant issues in the clustering techniques [29]. 2) having one more cluster besides the five clusters that we already have produces centroids that will be closer to some vectors than any other centroid among the existing centroids. Nevertheless, the results of the maximization we have here are all true results and can be used for further applications to identify unobserved passwords. Recalling the primary goal of this paper, which is developing a password metric based upon optimizing and maximizing centroids according to a set of weak passwords, we still can use any centroids we produced from the previous training sessions. We believe that using the maximized centroid C_{x5} would produce the strongest password meter in comparison with the meters from C_{x3}, C_{x4} or C_{x6}. Whereas developing a meter according to C_{x3} would be the most tolerant among all other meters that we have from the maximization process.

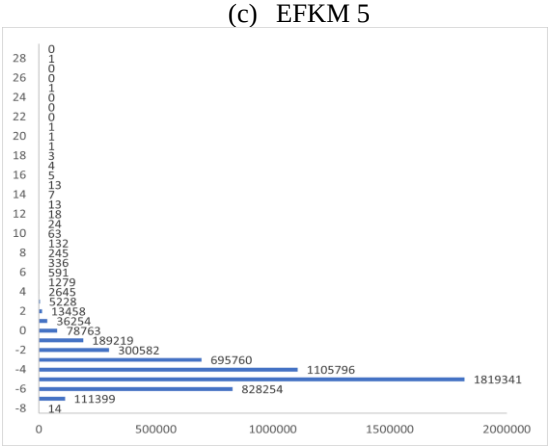
(a) EFKM 3



(b) EFKM 4



(c) EFKM 5



(d) EFKM 6

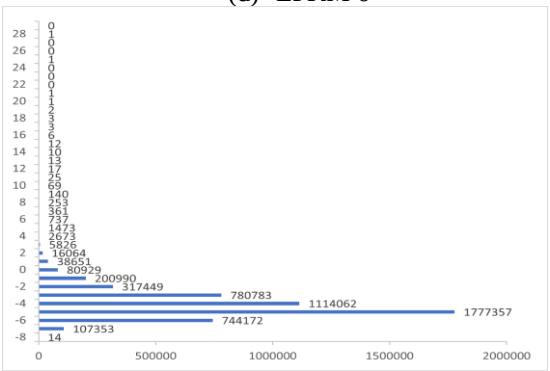
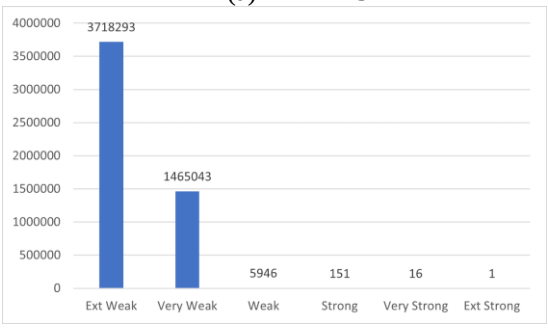
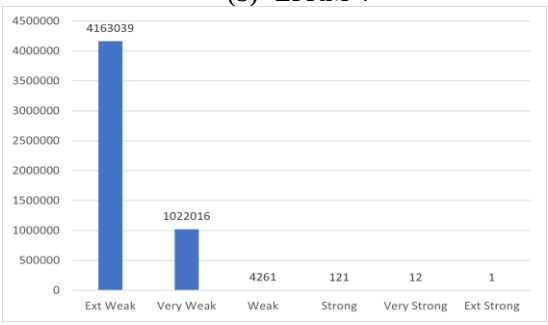


Figure 6. Passwords Distance-Based Distributions using EFKM Metric

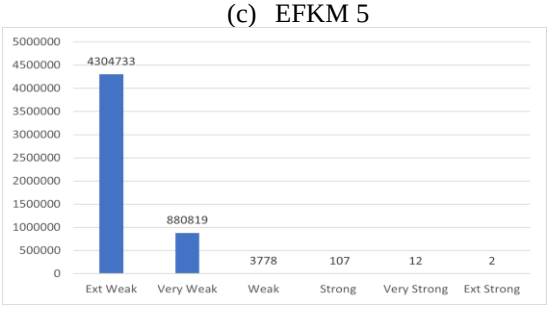
(a) EFKM 3



(b) EFKM 4



(c) EFKM 5



(d) EFKM 6

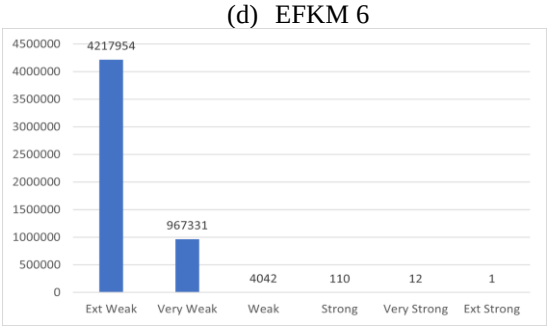


Figure 7. Passwords Categories using EFKM Metric

6. EFKM Meter Validity

To show the validity of EFKM metric as an efficient password metric, we build our argument on three bases: 1- EFKM metric application on OWASP passwords. 2- EFKM metric application on leaked passwords with comparison to entropy-based metrics application. 3- EFKM metric statistical correlation with entropy-based metrics.

1- EFKM metric application on OWASP passwords:

To evaluate the performance of the EFKM metric, we applied it on the OWASP dataset. Since the OWASP contains weak passwords only, the EFKM metric showed an excellent performance as it successfully recognized all the passwords in OWASP dataset as weak passwords, and this reflects the quality and the efficiency of our EFKM meter. Table 2 below shows that the average distance between OWASP passwords and the maximized quintuple under consideration is negative, so most OWASP passwords have definitely negative distances, and therefore they are weak passwords and none of them holds any level of strength. The averages show that all the metrics are good meters, regardless of what the number of clusters is. However, we consider the EFKM meter with 5 clusters to be the best one.

Table 2. OWASP Passwords Distances

	EFKM3	EFKM4	EFKM5	EFKM6
MIN	-7.962	-8.433	-8.646	-8.527
MAX	1.011	0.541	0.327	-4.525
AVG	-5.342	-5.812	-6.026	-5.259

2- EFKM metric application on leaked passwords:

The key idea of this validation of EFKM metric is contrasting figure 2 and figure 3 from one side with entropies figures from the other side. This contrasting shows that EFKM metric has preserved the distributions of leaked passwords intact. In other words, the passwords we labeled as weak in entropy-based metrics remained weak in EFKM metric. Moreover, EFKM metric was even harder than entropy-based metrics.

3- EFKM metric statistical correlation with entropy-based metric:

Table 3. Pearson's Correlations of the Metrics

	NIST	PURE	UNIFORM	EFKM
NIST	1.00	0.78	0.91	0.81
PURE	0.78	1.00	0.79	0.52
UNIFORM	0.91	0.79	1.00	0.70
EFKM	0.81	0.52	0.70	1.00

Table 3 shows the Pearson's correlations between different metrics that we used. Note that they are all positive values grater that 0.5, meaning that EFKM metric is positively correlated with the other metrics. It is even close to 1 in case of NIST and UNIFORM. This value shows a very high positive correlation with EFKM. This supports and concludes our argument on the validity of EFKM metric for password strength. Interestingly, the EFKM metric correlates to other metrics with the same value regardless of the number of clusters.

7. Conclusions

In this paper we introduced a password strength metric using the EFKM clustering algorithm. We trained the EFKM on the OWASP dataset that comprises 10002 weak passwords. After that, we maximized the optimized centroids to develop the required password strength metric. We tested the validity of the meter using two datasets: the training dataset (OWASP) and the leaked passwords dataset that we collected from the GitHub website and contains 5189451 leaked passwords.

Our EFKM metric could recognize all the passwords from the OWASP as weak passwords only. Regarding the leaked passwords, the metric recognized almost the entire set as weak passwords. As the passwords in the leaked dataset have no labels, we used other meters (NIST, UNIFORM and Pure Entropies) besides the EFKM-based meter for comparison purposes. We found that the results of the EFKM-based metric and the entropy-based meters were consistent. Hence the EFKM-base metric demonstrated its validity as an efficient password strength checker.

References

[1] A. R. L. Reyes, E. D. Festijo, R. P. Medina, "Securing one time password (OTP) for multi-factor out-of-band authentication through a 128-bit blowfish algorithm," International Journal of Communication Networks and Information Security, Vol. 10, No. 1, pp. 242-247, 2018.

[2] C. Herley, V. P. Oorschot, "A research agenda acknowledging the persistence of passwords," IEEE Security & Privacy, Vol. 10, No. 1, pp. 28-36, 2011.

[3] M. N. Todd, "An investigation of machine learning for password evaluation," MSc thesis, Arizona State University, USA, 2016.

[4] N. M. Aljaffan, "Password security and usability: from password checkers to a new framework for user authentication," PhD thesis, University of Surrey, UK, 2017.

[5] K. Sivapriya, L. R. Deepthi, "Password strength analyzer using segmentation algorithms," 5th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, pp. 605-611, 2020.

[6] M. M. Taha, T. A. Alhaj, A. E. Moktar, A. H. Salim, S. M. Abdullah, "On password strength measurements: password entropy and password quality," International Conference on Computing, Electrical and Electronic Engineering (ICCEEE), Khartoum, Sudan, pp. 497-501, 2013.

[7] G. Hu, "On password strength: a survey and analysis," Software Engineering, Artificial Intelligence, Networking and Parallel / Distributed Computing, pp. 165-186, 2018.

[8] D. Florencio, C. A. Herley, "A large-scale study of web password habits," International conference on World Wide Web, Alberta, Canada, 2007.

[9] K. Solic, H. Ocevcic, D. Blazevic, "Survey on password quality and confidentiality," Automatika, Vol. 56, No. 1, pp. 69-75, 2015.

[10] S. Komanduri, R. Shay, P. G. Kelley, M. Mazurek, L. Bauer, N. Christin, L. F. Cranor, S. Egelman, "Of passwords and people: measuring the effect of password-composition policies," The ACM CHI Conference on Human Factors in Computing Systems, Vancouver, Canada, 2011.

[11] R. Shay, P. G. Kelley, S. Komanduri, M. L. Mazurek, B. Ur, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, "Correct horse battery staple: exploring the usability of system-assigned passphrases," The eighth Symposium on Usable Privacy and Security (SOUPS), Washington DC, USA, 2012.

[12] R. Shay, S. Komanduri, P. G. Kelley, P. G. Leon, M. L. Mazurek, L. Bauer, N. Christin, L. F. Cranor, "Encountering stronger password requirements: user attitudes and behaviors," The sixth Symposium on Usable Privacy and Security (SOUPS), Redmon, USA, 2010.

[13] Z. Zheng, H. Cheng, Z. Zhang, Y. Zhao, P. Wang, "An alternative method for understanding user-chosen passwords," Security and Communication Networks, Vol. 2018, p.p. 1-12, 2018.

[14] M. Yildirim, I. Mackie, "Encouraging users to improve password security and memorability," International Journal of Information Security, Vol. 18, No. 6, pp. 741–759, 2019.

[15] M. Grimes, J. Proudfoot, P. Benjamin, "Improving password cybersecurity through inexpensive and minimally invasive means: detecting and deterring password reuse through keystroke-dynamics monitoring and just-in-time fear appeals," Information Technology for Development, Vol. 20, No. 2, pp. 196-213, 2013.

[16] E. F. Gehringer, "Choosing passwords: security and human factors," IEEE 2002 International Symposium on Technology and Society (ISTAS'02), Raleigh, NC, USA, pp. 369-373, 2002.

[17] K. Siau, Y. Ma, N. Twyman, "Cybersecurity: personal information and password setup," MWAIS Conference, St. Louis, Missouri, USA, pp. 1-6, 2018.

[18] Z. Liu, Y. Hong, D. Pi, "A large-scale study of web password habits of Chinese network users," Journal of Software, Vol. 9, No. 2, pp. 293-297, 2014.

[19] B. Ur, P. J. Kelley, S. Komanduri, J. Lee, M. Maass, M. L. Mazurek, T. Passaro, "How does your password measure up? the effect of strength meters on password creation," USENIX Security Symposium, Bellevue, USA, pp. 65-80, 2012.

[20] J. Galbally, I. Coisel, I. Sanchez, I., "A new multimodal approach for password strength estimation - Part I: Theory and algorithms," IEEE Transactions on Information Forensics and Security, Vol. 12, No. 12, pp. 2829-2844, 2017.

[21] B. Ur, "Supporting password-security decisions with data," Ph.D. Dissertation. Carnegie Mellon University, 2016.

[22] D. Wang, D. He, H. Cheng, P. Wang, "Fuzzy PSM: a new password strength meter using fuzzy probabilistic context-free grammars," 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 595-606, France, 2016.

[23] M. Golla, M. Dürmuth, "On the accuracy of password strength meters," The 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, Canada, pp. 1567–1582, 2018.

[24] S. L. Alqahtani, H. Yuan, P. Rusconi, "Human-generated and machine-generated ratings of password strength: what do users trust more?," EAI Endorsed Transactions on Security and Safety, Vol. 18, No. e1, pp. 1-16, 2020.

[25] A. Singh, S. Raj, "Securing password using dynamic password policy generator algorithm," Journal of King Saud University - Computer and Information Sciences, pp. 1-5, 2019.

[26] Open Web Application Security Project, https://owasp.org/

[27] H. Migdady, M. M. Al-Talib, "An enhanced fuzzy K-means clustering with application to missing data imputation," Electronic Journal of Applied Statistical Analysis. Vol. 11, No. 2, pp. 674–686, 2018.

[28] W. Burr, D. F. Dodson, W. Polk, "Electronic authentication guideline," NIST Special Pub 800-63, 2006.

[29] I. Khan, Z. Luo, J. Z. Huang, W. Shahzad, "Variable weighting in fuzzy k-means clustering to determine the number of clusters," IEEE Transactions on Knowledge and Data Engineering, Vol. 23, No. 9, pp.1838-1853, 2019.