Name: Akshada Rajpure
Roll no:5878
Class: SYCS

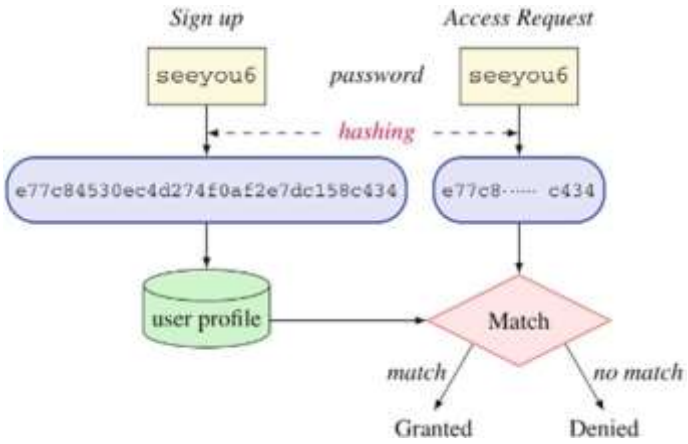
On Password Strength: A Survey and Analysis

Abstract: Password has been a predominating approach for user authentication to gain access to restricted resources. The main issue with password is its quality or strength, i.e. how easy (or how hard) it can be "guessed" by a third person who wants to access the resource that you have access to by pretending being you. In this paper, we review various metrics of password quality, including one we proposed, and compare their strengths and weaknesses as well as the relationships between these metrics. We also conducted experiments to crack a set of passwords with different levels of quality. The experiments indicate a close positive correlation between the difficulty of guessing and the quality of the passwords. A clustering analysis was performed on the set of passwords with their quality measures as variables to show the password quality groups.
Keywords Password quality · Entropy · Levenshtein distance · Hashing · Password cracking

Introduction

Online security has been a major concern since the time when the Internet became a necessity for the society, from business activities to everyday life of ordinary people. A fundamental aspect of online security is to protect data from unauthorized access. The most commonly used method for doing this is to use password as part of the online access process. Password is a secret character string only the user knows and its hashed code is stored on the server that provides access to the data. When the user requests for data access, he enters the password along with other identification information, such as user name or email. A message digest cypher (hash) of the password is computed and the hashed code is transmitted to the server that matches

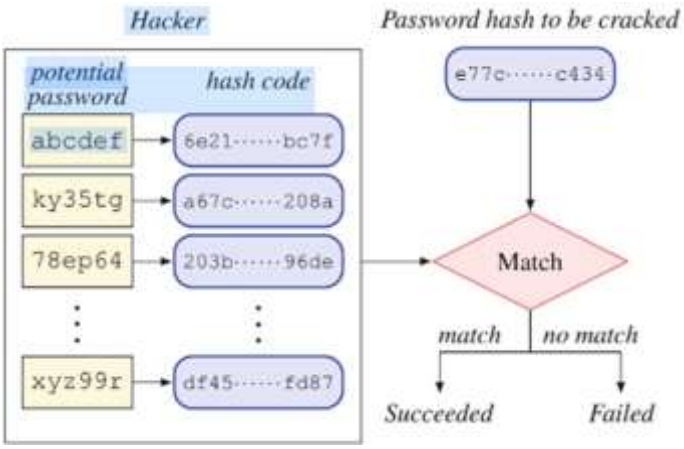
Fig. 1 Password mechanism for resource access request



Name: Akshada Rajpure
Roll no:5878
Class: SYCS

Fig. 2 Hacking the hash

code of a password



code previously stored in its database. If a match is found, the user is assumed to be the one who claims who he is, and access is granted; otherwise access is denied, as illustrated in Fig. 1.
Formally, a hash function f maps a password (character string) p to a hash code (cypher) h:

$$f(p) = h \quad (1)$$

where f is irreversible, i.e. function f^{-1} does not exist.
Since the hashing function is irreversible, the only way a third person (a hacker) can guess the password is to try a possible character string to see if its hash code matches the hash code of the actual password. If no match, the hacker may try another string, and continues this process (for off-line attack) until a match is found (the password is cracked) or the hacker gives up without success. The idea of cracking a password is shown in Fig. 2, where the hacker tries a sequence of potential password p1, p2,..., pk to generate hash codes h1, h2,..., hk, and to see if hk = h, where h is the given hash code to be cracked. For online attack, a "three strikes" type of rule may prevent the hacker to try more than 3 times, though.
The problem for the hacker to figure out is how to select strings p1, p2,..., pk as

potential passwords. There are various approaches for doing so, all based on guessing. From the user's point of view, the critical point to the secrecy of the password

is make it very hard for hackers to guess the password from the hash code. This is the issue of password quality (or strength).

2 Password Cracking Methods

Each password p is hashed using an encryption function f to generate its hash code h as defined in Eq. (1). Since only h (rather than p) is stored in the database on the resource server, the task of cracking a password is to use a method c such that $c(h) = p$. Note that c is not f^{-1} that doesn't exist. So, the hacker needs to figure out what method c to use such that he has a better chance to succeed finding p.

Name: Akshada Rajpure
Roll no:5878
Class: SYCS

Commonly used methods for cracking passwords include brute-force attack, dictionary attack, and some variations of these with time-space tradeoff considerations.
2.1 Brute-Force Attack
Let p = a1a2 ...al

, ai ∈ Σ be a password of length l, where Σ is an alphabet of character set and N = |Σ|. Given the hash code h = f(p), brute-force attack is to try each

possible string s = x1x2 ...xl

, xi ∈ Σ until f(s) = h.

The amount of time t to crack a password is proportional to the number of possible strings:

t ∈ Θ(Nl
) (2)

where Nl is the size of string pool the possible passwords are drawn from the alphabet Σ. That is, the time complexity of brute-force attack is polynomial of N and

exponential of l. In general, the alphabet Σ is one of the basic sets or combination of them. The basic sets as given in Table 1.
Most passwords are from lowercase letters (L) only. In this case, for a password of length 6 (minimum length required by most service sites), there are 266 ≈ 308.9 × 106 possible strings to try. If we can make 100,000 calls to the crypt function f per second, it will take 3089 s, or 51.5 min, to exhaust all possible strings from L. For a larger alphabet, say D ∪ L, a 6-char password will need over 6 h to crack, while a 7-char password needs over 9 days, and 11 months for a 8-char password.
It is practically infeasible to use brute-force approach to crack a password of length 7 or longer with a larger Σ. It is the basic reason why most service sites require passwords to be at least 8 characters long, with at least an uppercase letter, and a digit or a special character, in addition to lowercase letters. This will make N = (62 or 94) and l ≥ 8.

Password Quality Measures

Analysis of password strength has been an active area for research and practice for a long time. The focus of these work is on the metrics of password strength and evaluation of these metrics. We shall survey several metrics for password quality includeing complexity that we propose here.

Password Quality Indicator

Most users are aware of dictionary attack and avoid using dictionary words for passwords. However, users want passwords easy to remember, so they tend to use a word

and make small changes to it. With this consideration, methods of dictionary attack

also adopted various word-mangling rules to match a password with words in dictionaries. So, the strength of a password not only considers if the password is in dictionaries, but also should measure how easy (or hard) to correct the "spelling errors"

in the password so it can match some words in the dictionaries. This is a commonly measure as a linguistic distance between the password and a dictionary word. The very basic linguistic distance is the Levenshtein distance (or edit distance) that is the minimum number of editing operations (insert, delete, and replace) needed to transfer one word to another. This idea was used in the password quality index (PQI) metric proposed in [13] and refined in [14].
The PQI of a password w is a pair $\lambda = (D, L)$, where D is the Levenshtein distance of w to the base dictionary words, and L is the effective length of p, which is defined as

$$L = m \times \log_{10} N \quad (6)$$

where m is the length of w and N is the size of the charset where the characters of w is drawn from.
The effective length is the length calculated in a "standardized" charset, the digit set D given in Table 1. The idea behind this is that a password (e.g. k38P of length 4) drawn from multiple charsets ($D \cup L \cup U$) is just as hard to crack (or, has about the same number of possible candidates to crack) as another password (e.g. 378902 of length 6) from only the digit set D.
It is seen that the effective length of a password given in (6) is essentially the same as the entropy value in (5), just off by a constant factor $\log_2 10$. Both consider the password's length as well as the size of the charset.
With the PQI measure, The quality criterion given in [13] states that a password is of good quality if $D \geq 3$ and $L \geq 14$.

Password multi-checker output for password$1

Service	Strngth score	
Apple	Moderate	2/3
Dropbox	Very weak	1/5
Drupal	Strong	4/4
eBay	Medium	4/5
FedEx	Very weak	1/5
Google	Fair	3/5
Microsoft (v3)	Medium	2/4
PayPal	Weak	2/4
Skype	Poor	1/3
Twitter	Perfect	6/6
Yahoo!	Very strong	4/4

Password Strength Test

To further evaluate the password strength metrics, we experimented cracking the passwords in the small data set D3 with 127 passwords that are manually selected to

include strengths as various levels.

Clustering of Passwords

As another evaluation of the strength metrics is to do clustering of the passwords to see if the weak passwords are grouped together and the strong ones are grouped in another cluster. In addition to the 4 variables (entropy, NIST, complexity, and Levenshtein distance), we also used two more variables when we applied the clustering algorithm. The two variables are effective length and a variation of the Levenshtein distance, which calculates the Lavenshtein distance of sub-words in a password against dictionary words, rather than treating the password as a single word. We applied the K-means clustering algorithm [10] on the 127 passwords in the data set D3 with the 6 variables into 4 clusters representing four strength levels.

Conclusion

we reviewed the basic metrics for assessing password strength, including entropy, NIST entropy, password quality index, and Lavenshtein distance, as well as some password quality metrics developed at some popular service vendors. We proposed a password complexity metric that considers the composition patterns in a password in addition to the LUDS criteria. The correlations between these measures were analyzed using the maximum information coefficient (MIC) and Pearson coefficient. The resulting statistics show that most of the metrics are closely correlated except Levenshtein distance that may be a good measure for password quality besides the traditional parameters (length of password and size of charset).
There password strength metrics were evaluated by experiments that tried to crack the hashes of a small set of passwords to see if the difficulty of cracking a password is indeed related to the strength measures. The cracking tools used including techniques like brute force, transformation rules, dictionary attacks, and massive table look-up. Two types of results were obtained: password found or not (table look-up), and the length of time spent to discover the password (other techniques used in JtR). Results showed that the level of success cracking the passwords is highly positively related to the strength measures. It is pretty convincing from our experiments that the strength metrics are valid and can be used to assess the quality of user selected passwords. This was further validated by the clustering results.