World of Computer Science and Information Technology Journal (WCSIT)
ISSN: 2221-0741
Vol. 3, No. 6, 110-113, 2013

The Effect of Weight Factors Characters on Password Selection

Ghadeer Ali Shaheen, Prof. Dr. Ahmad Al Jaber, Prof. Dr. Alaa Al Hamami
Department of Computer Science,Amman Arab University
Amman, Jordan

Abstract— The world of today is ruled by the internet, where everyone from individuals to institutions stores their information on it. Passwords are one of the important things for any system , it have been used for a long time in many applications, such as logging into computer accounts , Email , banks , shopping online , transferring funds , accessing program , database , networks , portals dating and social networking sites all require passwords. Due to the limitation of human memory, people are inclined to choose easily guessable passwords that lead to severe security problems. In this paper, we will propose a new program that create strong password based on factors characters weight method, the length of password, diversity of its characters using Huffman coding compressing algorithm. As a result to the experimental work, we conclude that the proposed program provides the user with several password suggestions to select a safe, strong password and avoid any hacking programs or techniques guesses the password.

Keywords- Passwords; Hacking; Internet security; Password Meter; Huffman Coding.

I. INTRODUCTION

Life these days has become largely dependent on passwords for many purposes: logging into computer accounts, retrieving emails from servers, transferring funds, shopping online, accessing programs, databases, networks, web sites and even reading the morning newspaper online.

The problem of selecting and using good passwords is becoming more important every day. The importance of services that are provided through computers and networks increases dramatically and in many cases such services require passwords or other forms of security concerns.

In this paper, we will create an effective password through authentication system software based on a factors characters weight method, length of password, diversity of its characters using Huffman coding compressing algorithm.

II. RELATED WORK

There are interests in the field of Password and password development. A password is a secret word that is used for authentication, to prove identity or gain access to a resource. Passwords (watchwords) have been used since ancient times, in Roman military; it was the way in which they secure the passing round of the watchword for the night [1]. The UNIX system was the first to have a file that contained the actual passwords of all users. But that was insufficient; anyone could reach the password file, edit or modify it, even make a copy of this file.[2]

In the proposed model, we apply some of the passwords types and present a new authentication effective password software using Huffman coding algorithm.

A. Textual Password

The password may consist of English letters, numbers and symbols. Text passwords have been widely used for user authentication. Human-generated text-based passwords can be divided into three categories: Non-word passwords, Mixture passwords, Word passwords and passcodes [3].

B. Graphical password

The main difference to textual passwords is the use of a device with graphical input, the user enters the password by clicking on a set of images, specific pixels in images, or by drawing a pattern in a pre-defined and secret order [4].

C. Text and Graphical Passwords

Combining text and graphical passwords can be done by following two steps: In step one a user is asked for her user name and text password. In step two the user is presented with an image portfolio. , if both the text password and all graphical passwords were correct, he is granted account access. Otherwise, access is denied [5].

III. TOOLS AVAILABLE FOR PASSWORD STRENGTH CHECKING

Commercial tools available for password strength checking includes the Password Meter (Password Meter, 2008) and Microsoft password checker (Microsoft, 2008). These password meters use lexical rules.

The Password Meter which used by Google is a Java Script Function That checks the strengths of passwords with a well-defined algorithm.

It is based on dealing with a weighting method, and a weight is adopted for computing the strength of the password. The strength is decided based on the overall score which is determined using positive and negative weightages based on the scheme given in Table(1), the final score is capped with minimum of zero and a maximum of 100. The features that make the password strong are given more weightage and the features that weaken the password are given negative weightage.

TABLE 1 SCHEME OF WEIGHTS ASSIGNED

Additions	**Weight Assigned**
Number of characters in the password	Number of characters*4
Number of Lowercase characters	(length – number of lowercase characters) * 2
Number of Uppercase characters	(length – number of lowercase characters) * 2
Number of digits	(number of digits * 4)
Number of symbols	(symbol count * 6)
Number of Middle number /symbols	(number/symbol count * 2)
Deductions	
Characters only	- 1 * number of characters
Digits only	- 1 * number of digits
Number of repeated characters (n)	- (n (n –1))
Number of consecutive uppercase characters (n)	- (n * 2)
Number of consecutive Lowercase characters (n)	- (n * 2)
Number of sequential characters	- (n * 3)
Requirements (n)	- (n * 2)

The final score is the cumulative result of all bonuses and deductions, and the final score is capped with minimum of 0 and a maximum of 100 [6].

TABLE 2 PASSWORD CLASSIFICATION FOR THE FINAL SCORE.

Class	**Score**
Very Weak	Less Than 20
Weak	20 – 39
Good	40 – 59
Strong	60 – 79
Very Strong	Greater than 80

IV. HUFFMAN CODING

The most common way to represent characters and numbers in computing is by using the ASCII Code or Unicode. ASCII Code is based on a string of 8 bits where each bit can be either ‘1’ or ‘0’. Unicode Code is based on a string of 16 bits where each bit can be either ‘1’ or ‘0’. The advantage to these two systems is: when reading a file, it always reads 8 bits or 16 bits at a time for a single character.

But these coding schemes have disadvantage because some characters are more frequently used than other characters.

Huffman Coding can find the optimal way to take advantage of varying character frequencies in a particular file. Huffman Coding give less frequent characters longer codes, and more frequent character shorter codes [7].

V. SCENARIOS IMPLEMENTATIONS AND ANALYSIS

In order to test and measure the reliability of our new software, we present four scenarios; these scenarios explain the main functionalities that our software should do.

The four scenarios are:

(1) Password Generating.

(2) Finding the passwords’ strength by using Huffman Coding Checking algorithm.

(3) Generated passwords which obtained from Huffman Coding Checking algorithm will be checked again according to Password Meter checking algorithm.

(4) Providing the user with several suggestions to select a safe and strong password.

A. Algorithm Scenario One:

The user is going to enter his/her four categories of Characters (Lowercase, Uppercase, digit and symbol) in this scenario.

The program starts generating the password according to the following algorithm:

```
Password Generator Algorithm:
begin
proc findPermutations(elemints:Array,len:int)
permutationsNum=Math.pow(elements.length,len));
check();
end
begin
proc check()
permutations:Array;
while permutations.length < permutationsNum
perm:Array;
while ( perm.length < len )
 ind:int = Math.random() * elements.length;
 perm.push(elements[ind]);
 permstr:String=perm.join(','
 do if (permstr is not in permutations)
  permutations.push(permstr);
              end
```

Figure 1. Password Generator Algorithm

B. Algorithm Scenario Two:

In this scenario the program find the passwords' strength by using Huffman Coding Checking algorithm.

```
Huffman Coding Checking Algorithm:
 1.   Find Huffman Coding for each password.
 2.   Calculate the entropy for each password according to
      formula:
 Entropy = - ∑ pi log2 pi
 3.   Create five Huffman Coding Classifications:
 ( Very Strong, Strong, Good , Weak , Very Weak)
 4.   Distribute each password according to its entropy among
      these five groups.
 5.   Save the first 20 results from each Group.
```

Figure 2. Huffman Coding Checking Algorithm.

Huffman Coding Classification created according to the following Scores:

TABLE 3 HUFFMAN CODING GENERAL CLASSIFICATIONS

Class	Score
Very Weak	2.21 – 2.5
Weak	1.91 – 2.20
Good	1.61 – 1.90
Strong	1.31 – 1.60
Very Strong	1.00 – 1.30

C. Algorithm Scenario Three:

Generated passwords which obtained from Huffman Coding Checking algorithm will be checked again according to Password Meter checking Algorithm.

```
Password Meter Checking Algorithm:
1. Find the weight for each password by computing the overall
score which is determined in schemes:
- Number of characters in a Password.
- Number of lowercase characters.
- Number of uppercase characters
- Number of digits
- Number of symbols.
- Number of middle number/symbols.
- Character Only
- Digit Only
- Number of repeated Characters
- Number of consecutive uppercase characters
- Number of consecutive lowercase characters.
- Number of sequential characters.
- Requirements
2. Create five Password Meter Classifications
(Very Strong, Strong, good, Weak, Very weak).
3. Distribute each password according to its weight among these
five groups.
4. Save the first 20 results from each group.
```

Figure 3. Password Meter Checking Algorithm

D. Algorithm Scenario Four:

This Algorithm will provide the user with several suggestions to select a safe and strong password.

We did analysis for the result obtained in step three, it found the following results:

- ❑ Some results got the same positive class classifications in both rules.

According to Password strength basic tests, generated passwords satisfied the following points:

- Character type analysis: the generated passwords contain ¾ of the following character groups:

- Uppercase Letters.

- Lowercase Letters.

- Numbers.

- Symbols.

- Length distribution analysis: the length of generated passwords is eight characters which satisfied minimum password length.

- Common password analysis: the passwords generated randomly so we avoid the most common passwords.

To get a (strong score) both in Huffman coding and Password Meter, we must have the following points in generated password:

1. In Huffman Coding Checking, the generated password must contain three characters from different groups, example (3SwSw3Sw).
2. In Password Meter Checking, the generated password must be in the following distribution:

TABLE 4 STRONG PASSWORDS FORMULA

CCLDLLCL	Where: C: Capital letter (Different letters) S: Small letter (Different letters) D: Digit. L: Symbol
DDCLDLDC	
DCSCSDCS	
DCSCSDCS	
CSDDCSCS	
DDLCLCDC	
DDLSLSDS	
SSLDLLSL	
DDSLDLDS	
DSCSCDSC	

❑ Some positive results are very close to each other.

To get positive result very close to each other generated password must contain the following points:

1. In Huffman Coding checking the generated passwords must contain the following no of groups.

TABLE 5 POSITIVE RESULT IN HUFFMAN CODING CHECKING

Strong score	**Good score**
Three characters from different groups or two different characters from the same group, one from other group	Four characters from different groups or two different characters from the same group, two character from other two groups.

2. In Password Meter checking generated passwords will get positive score when it satisfies all additional criteria's.

❑ There are a lot of negative results.

To get very strong score in Huffman Coding the password must contain two characters from different groups with the same no of repetition, example (Z!!!Z!ZZ) which means that Password Meter criteria is not satisfied, so the score will be always very weak.

❑ There are no results in very strong password score.

To get very strong score both in Huffman Coding Checking and Password Meter checking generated passwords must contain the following points:

1. In Huffman Coding Checking, the generated password must be from two character groups with the same no of repetition, example: Z!!!Z!ZZ

To get very strong password in Password Meter, the password should satisfy its criteria's, and the password must contains 3/4 of the following items:
- Uppercase Letters
- Lowercase Letters
- Numbers
- Symbols

VI. CONCLUDING REMARKS AND FUTURE WORK

In this research, we were mainly concerned in creating a new authentication system program based on factors of the characters weight method, the length of password, diversity and repetitions of its characters using Huffman coding compression algorithm.

The program shows that getting a (strong score) both in Huffman coding and Password Meter, generated password must contains the following points:

1. In Huffman Coding Checking, the generated password must contain three characters from different groups.
2. In Password Meter Checking, the generated password must be with specific distribution:

As a Future work it will be mainly focused on:

- Modifying this approach by increasing the length of the password and check the relationship between the length and the strength of password in Huffman coding algorithm.
- Using Huffman Coding with other password strength checking tools.
-
- Create a new algorithm for generating password and password recovery.

REFERENCES

[1] Password (June 2008) http://en.wikipedia.org/wiki/Password [accessed 1/4/2011].

[2] Morris, R. and Thompson, K.(November 1979) Password Security : A Case History , Vol 22.

[3] Helkala, K. and Snekkenes, E. (July 2009). Password Generation and Search Space Reduction. Journal of Computers , Vol. 4, No. 7.

[4] Monrose, F. and Reiter, M. (August 2005) Graphical Passwords. USA: O'Reilly Media.

[5] Oorschot, V. and Wan, T. TwoStep : An Authentication Method Combining Text and Graphical Passwords.

[6] Jamuna, K. , Karpagavalli, S. and Vijaya, M. , (November 2009),A Novel Approach For Password Strength Analysis through Support Vector Machine. International Journal of Recent Trends in Engineering , Vol. 2 , No. 1.

[7] Huffman Coding. www.cs.ucf.edu/~dmarino/ucf/cop3503/ .../HuffmanCoding01.doc [accessed 8/03/2011].