Password Strength Checker – ENIGMA

Minor Project

IBM Career Education

Disclaimer

This Software Requirements Specification document is a guideline. The document details all the high level requirements. The document also describes the broad scope of the project. While developing the solution if the developer has a valid point to add more details being within the scope specified then it can be accommodated after consultation with IBM designated Mentor.

INTRODUCTION

The purpose of this document is to define scope and requirements of a Password Strength Checker – ENIGMA for a leading bank who wanted to provide enhanced password security to its customers. Currently the bank's internet banking website did not offer password strength check thereby making bank accounts vulnerable to password hacking using brute force or dictionary attacks.

The proposed system will provide an effective way to determine password strength whenever customer changes his/her password. If the password chosen by the customer is weak and susceptible to password attacks, the system will provide instant feedback and advise the customer to choose stronger password.

This document is the primary input to the development team to architect a solution for this project.

System Users

The entire customer base of the bank will primarily use the password strength checker, ENIGMA.

Assumptions

1. The password dictionary will be uploaded using a CSV file. Such a file will be a single column CSV file containing the list of commonly used passwords.
2. Whenever a new password dictionary is uploaded, the previously stored dictionary is cleared and a new dictionary is uploaded in place of the old one. All the necessary computations are carried out again for the newly uploaded password dictionary.

REQUIREMENTS

ENIGMA will determine the strength of a password on client side (i.e. without needing to make a round trip to the server) on the basis of the following rules:

1. Password length has to be at least 8 characters or more.
2. It must include alphabets, numbers and at least one special character.
3. Password should not be present in the dictionary of commonly used passwords.

Basic System Operation

The system will leverage "bloom filter" for performing a lookup in to the password dictionary. Bloom Filter is an extremely memory efficient probabilistic data structure for a very quick negative look up. In other words, it can determine that the password in question is "not" in the dictionary of commonly used passwords.

The administrator uploads the dictionary in the system. Upon successful upload,

ENIGMA builds the “bit vector” for the passwords in the dictionary. During the password change operation, this bit vector travels to client side in form of a JSON. The client side Java Script performs the negative look up using this bit vector.

During password change, user is presented 3 fields viz. current password, new password, and very password. The password strength check will be triggered on the exit of new password field.

About Bloom Filter

Bloom Filter is a probabilistic data structure. It is in form of a “bit vector” or an array of bits with all bits initialized to “0”. To add an element, it uses 2 (or more) hash functions that map the element to 2 (or more) of the array positions with a uniform random distribution. These mapped bit positions are set to “1”.

To perform a look up for an element, find the hash of that element using the same set of hash functions used during addition to get 2 (or more) array positions. If any of the bits at these positions are 0, then the element is not in the set.

Therefore, selection of hash functions will be critical, as the same set will be implemented in Java for server side operation to build the bit vector; and in Java Script to perform the negative look up on that bit vector.

The memory efficiency results from the bit vector usage. It will not require several bytes to store each password. For example in form of a bit vector, 10000 passwords can fit in to an array of about 1.5KB.

DEVELOPMENT ENVIRONMENT

ENIGMA will be developed as a web application using Java/JSP and DB2 database. Eclipse will be used as the IDE for the same. You may consider using a JavaScript framework like Prototype. Please refer to textbooks on data structures in your library for Bloom Filters & Hash Functions. You may also refer to http://www.serve.net/buz/hash.adt/java.001.html URL for concepts on hashing and to http://en.wikipedia.org/wiki/Bloom_filter URL for concepts on bloom filter.