International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 08 | Aug - 2021 ISSN: 2582-3930

DPPG: A Dynamic Password Policy Generation System

BIT Ballarpur Gondwana University

Madhuri Wasnik,
Ruchita Chaudhari,
BE Scholar
B.I.T.Ballarpur , Chandrapur

Gayatri Pachare ,
awanti Deshmukh
BE Scholar
B.I.T.Ballarpur , Chandrapur

Prof. Saraswati B.Sahu
Assistant Professor
B.I.T.Ballarpur , Chandrapur

Abstract- to keep password users from creating simple and common passwords, major websites and applications provide a password-strength measure, namely a password checker. While critical requirements for a password checker to be stringent have prevailed in the study of password security, we show that regardless of the stringency, such static checkers can leak information and actually help the adversary enhance the performance of their attacks. To address this weakness, we propose and devise the Dynamic Password Policy Generator, namely DPPG, to be an effective and usable alternative to the existing password strength checker. DPPG aims to enforce an evenly-distributed password space and generate dynamic policies for users to create passwords that are diverse and that contribute to the overall security of the password database. Since DPPG is modular and can function with different underlying metrics for policy generation, we further introduce a diversity-based password security metric that evaluates the security of a password database in terms of password space and distribution. The metric is useful as a countermeasure to well-crafted offline cracking algorithms and theoretically illustrates why DPPG works well.

Keywords – Dynamic Password, Authentication, ARP, Sniffer;

1. INTRODUCTION

TEXT-BASED passwords have been used widely in both online and offline applications for decades. Since passwords are personal and portable, they are not likely to be replaced in the foreseeable future [1]. However, the phenomenon that people choose simple passwords and reuse common passwords [2] has raised great security concerns as such passwords are vulnerable to offline cracking attacks. To make things worse, a number of password leak incidents [3]– [6] have happened recently and frequently. Large datasets of leaked passwords can greatly enhance attackers' capability in conducting training-based password attacks, thus posing significant threats on password security. On the other hand, the password strength checker itself can be a vulnerability, which has not been studied in previous research. By defining a set of password creation policies and showing users password strength scores, password checkers can exert a strong bias on password characteristics, especially when the policies and scoring

The most direct and pervasive protective mechanism used by major websites and applications is the password strength checker [7], which evaluates the strength of passwords proactively during user registration. While the goal is to guide users to create strong passwords, in previous work [8]–[10], the lack of accuracy and consistency in the strength feedback has been widely observed and examined. That is, existing checkers do not demonstrate effective or uniform characterization of strong passwords. Furthermore, the space for the rules and policies of the checkers to be stringent is very limited as researchers have shown that the complexity of a password is a trade-off with the usability [11] .

2 COMMERCIAL PASSWORD CHECK

Traditional password policies have become less popular as the more user-friendly password strength checkers become widely adopted by major websites and software. The main reason is that good password policies can easily be too stringent to use, while password strength checkers push users to create “strong” passwords subtly. However, most of the existing research only evaluates the effectiveness and helpfulness of the password strength checkers.

A .Datasets, Checkers, and Crackers

Table I lists the 5 datasets that add up to around 81 million passwords. The datasets are leaked from several incidents [13], [14] where attackers acquire passwords by online attacking techniques. Although the password data were leaked illegally, it has been once made publicly available and used widely

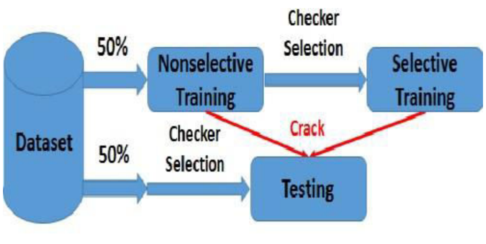


Fig 1: Attack-based Evaluation Model

In password research for benevolent purposes. In our study, we use the passwords for research only without attempting to verify them. To obtain a collection of usable password strength checkers and cracking algorithms, we conduct our experiments with PARS [10]. Due to the space limitation, we only present two checkers listed in Table II. Other checkers showing consistent results are available on [15]. Bloomberg is a popular English business and news forum and QQ is a well-known Chinese portal providing numerous web services. According to evaluations in [10], [12], they provide relatively accurate and consistent feedback to users. There are 4 levels of password strength in both password checkers to make them comparable, and the highest rating is “strong” in common.

B. Threat Model:

Take Your Checker, Crack Your Passwords From an attacker’s perspective, we evaluate quantitatively how existing commercial password checkers can be used to enhance offline password attacks. We are particularly interested in the pool of “strong” passwords because intuitively users trust the strength feedback and create passwords that have better ratings. In our threat model, we assume an attacker aims to crack a target set of password hashes leaked from a website which uses a password strength checker. This means that the hashed passwords can have different strength ratings 1 . We also assume the attacker has access to the checker and obtained another dataset of plain text passwords leaked from other websites as prior knowledge, which is used to train the password crackers. Since the attacker does not know the correlation between the plain text and the hashed passwords, a straightforward

3 DYNAMIC PASSWORD POLICY GENERATOR

One could argue that a potential solution to the password checker limitations is to have better web technologies to hide the policies and detect malignant password strength querying. However, it can result in delay in strength feedback and high falsepositive rates in detection. Further, it does not resolve the fundamental bias in password distribution. Therefore, we take another approach to the problem and explore the feasibility of providing dynamic password policies to users. Considering usability, rather than forcing all users to create

International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 08 | Aug - 2021 ISSN: 2582-3930

extremely complex passwords, we focus on the overall strength of the.

A.Overview DPPG is a diversity-based and database-aware application that generates password creation policies dynamically for the users. Instead of purely focusing on the complexity of candidate passwords, DPPG enforces a baseline complexity on the passwords (e.g., more than 6 characters long) to protect them from simple attacks, e.g., dictionary, brute-forcing. However, more focus is put on protecting the password distribution within a database by preventing aggregation of similar passwords that form a characteristically biased distribution.

B. Two Modes: Explore and Exploit

In order to intelligently generate password policies based on the current password distribution, DPPG maintains a global characteristics frequency map and a history of generated password policies3 that can approximate the current password distribution. There are two modes for DPPG to expand the usable password space and balance the password distribution. The exploration mode mainly aims to expand the password space by actively introducing new characteristics 3No plain text passwords.

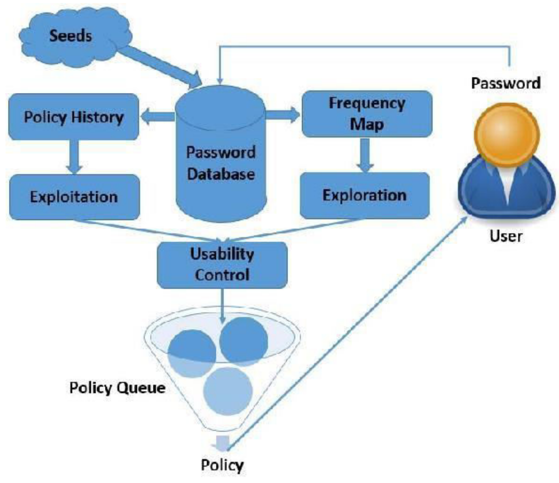


Fig. 3. Dynamic Password Policy Generator

As shown in examples below.

Include the character(s): ‘v’, ‘Z’

Avoid the character(s): a, s, e

Use the structure: LLLLLUUS

Number of characters: 8 to 12 (inclusively) Number of character types: 4

Number of alternations: 3 to 4 (inclusively)
Include the character(s): ‘?’, ‘U’, ‘)

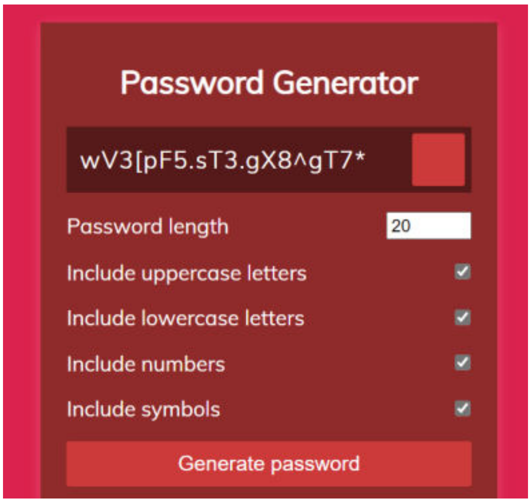
Visual studio

1) Hardware Requirements(minimum requirement)
Main processor Intel core
Hard disk capacity 300gb
Ram 2gb
Processor speed 512mhz2)
2)Software Requirement(minimum requirement)
Operating system: Window7/8/10
Above
Front End: HTML5, CSS3,BOOSTRAP4
JAVASCRIPT
Database: PHP / Mysqli
IDE: Visual Studio
Code / Sublime Text

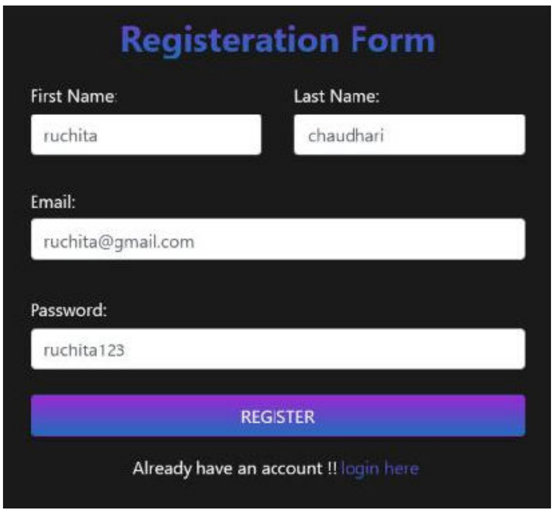
Random Password Generation With Algorithm

In this article we are going to create a random password generator, which will be highly flexible in terms of length of the password and the used characters set. The algorithm will be implemented in Scilab and also as an in-page application.
Passwords are critical for cyber security. A good password must have at least 8 random characters in length and should contain lower case letters, upper case letters, numbers and at least one special character

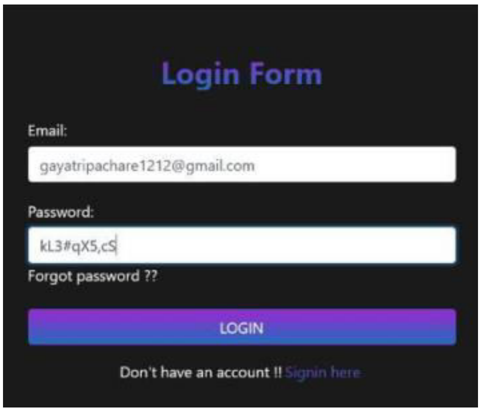
Password Generator



Registration form Random Password



Login Form Random Password



MD5 Hash Function

Encrypt passwords

The password hash() function creates a new password hash of the string using one of the available hashing algorithm. It returns the hash that is currently 60 character long, however, as new and stronger algorithms will be added to PHP, the length of the hash may increase. It is therefore recommended to allocate 255 characters for the column that may be used to store the hash in database.

The following algorithms are currently supported when using this function:
PASSWORD_DEFAULT
PASSWORD_BCRYPT
PASSWORD_ARGON2I
PASSWORD_ARGON2ID
Additional options can be passed to this function can be used to set the cost of encryption, the salt to be used during hashing, etc in the $options array.
Output:
Generated hash: $2y$10$7rLSvRVyTQORapkDOqmkhetjF6H9lJHngr4hJMSM2lHObJbW5EQh6

Decryption of the password

To decrypt a password hash and retrieve the original string, we usethe password verify() function.

The password_verify() function verifies that the

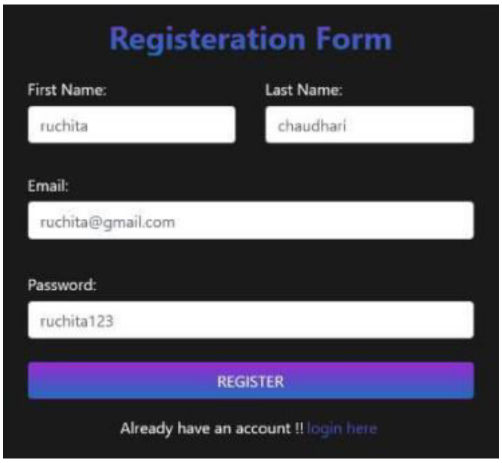
International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 08 | Aug - 2021 ISSN: 2582-3930

given hash matches the given password, generated by the password_hash() function. It returns true if the password and hash match, or false otherwise.

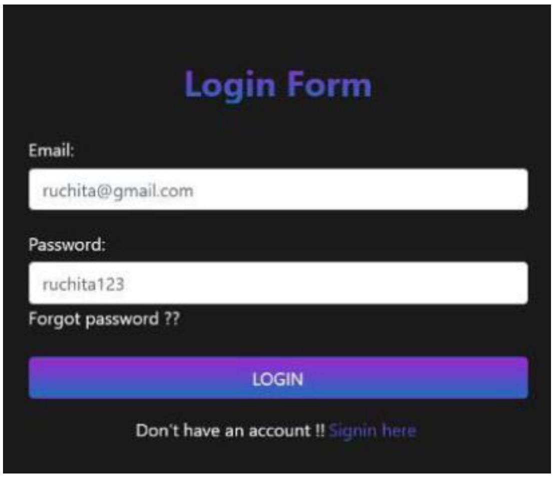
Output:

Password Verified!

MD5 Hash algorithm of Registration form

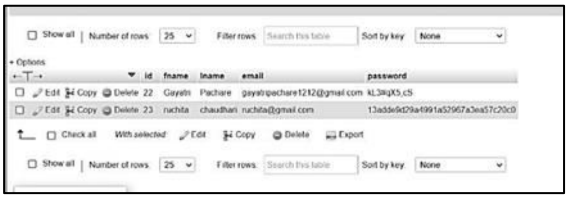


Login Form MD5 Hash Algorithm



Result

CONCLUSION



In this paper, we study the password space and distribution to understand password dataset security better. Due to the limitation of existing strength measuring mechanisms, we propose a new and usable alternative based on an effective diversity metric to better protect passwords from offline cracking attacks.

DPPG generated passwords though have a good memorisable rate, the user gets restricted in choosing passwords and not easy at every instance of password. People can argue that a potential solution to the password checker limitations is to have better web technologies to hide the policies. Thus it is a very efficient and easy to use application The checkers can be leveraged by the attackers easily to select training data that are similar to the target passwords.

AKNOWLEDGMENT

With immense pleasure and great respect I take this opportunity to express my deep sense of gratitude toward my guide, Prof. PQR, Assistant Professor, Department of Computer Science and Engineering, B.I.T Ballarpur for invaluable guidance, inspiration, constant encouragement and motivation throughout the project work

REFERENCES

[1] C. Herley, P. C. Oorschot, and A. S. Patrick, "Passwords: If we're so smart, why are we still using them?" FC, 2009.

[2] D. Florencio and C. Herley, "A large-scale study of web password ˆ habits," WWW, 2007.

[3]http://www.adeptusmechanicus.com/codex/jtrhcmkv/jtrhcmkv.php.

[4]http://www.zdnet.com/blog/security/chinese-hacker-arrested-forleaking-6-millionlogins/11064.
[5]"Yahoo!passwordleakege,"

http://www.cnet.com/news/yahoospassword-leak-what-you-need-to-know-faq/.