Password Checker∗

CPSC 4240 Project

Rajat Sethi  
 CECAS  
 Clemson University  
 Clemson SC USA  
 sethi@clemson.edu

ABSTRACT

There exist several methods to crack passwords, the most common being wordlist comparison, brute-force algorithms, and social engineering. The goal of this project is to create a program that verifies the strength of a password against these common methods of attack.

With wordlists, the easiest method of verification is a simple binary search. In this project, the program would iterate through major known wordlists like rockyou.txt and search for the user’s password. If the password is not found in these wordlists, then it passes the “wordlist” test.

With brute-force algorithms, there are a few methods of approach. Ultimately, for this project, the program would attempt a brute-force by testing every possible string and record how long the attack took. If the brute-force takes less time than the maximum allotted frame, then the password is unsafe. Otherwise, it passes the “brute-force” test.

For the scope of this project, “Social Engineering” was not included. While social engineering occurs on a case-by-case basis, it is worth looking into possible tests that check the “human” side of creating passwords.

The final program created for this project was able to successfully test wordlists in an efficient manner. In future endeavors, more work needs to be done for brute-force algorithms and social engineering.

CCS CONCEPTS

• Security and privacy • Security services   • Privacy-preserving protocols

KEYWORDS

Password, wordlists, brute-force, verification

1 Introduction and Objectives

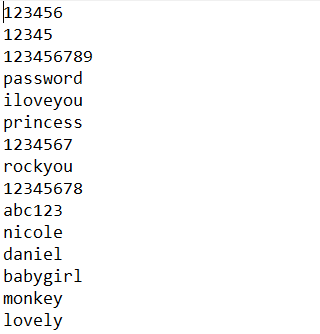
One of the largest issues in cybersecurity is the unsafe password. Many passwords are either too common, small, or repetitive for use, which makes them susceptible to several types of attack. This includes, but is not limited to, dictionary attacks with wordlists, brute-force attacks, and social engineering attacks. The goal of this endeavor is to create a system that can check passwords and assert that the password is virtually uncrackable to these common attack methods.

2 Implementation

The original goal of this project was to use C++ and Bash to test passwords. C++ would have custom methods while Bash would use tools that already existed. However, due to time constraints, only the custom C++ code was used. This program would take an indefinite amount of possible passwords from the user and examine them against a wordlist test and brute-force test.

2.1 Wordlist Examination

In the past, several companies have had their databases leaked and released to the public. The most prominent of these leaks was from “RockYou”, a company that created widgets for social media companies like Facebook and MySpace. In 2009, a database with over 32 million unencrypted passwords was leaked to the public.[1] Since the attack, that list of passwords has been organized into a simple text file, “rockyou.txt”, and now comes with distributions of Kali Linux.



**Figure 1: A Sample of rockyou.txt**

For this project, the password is cross-referenced with “rockyou.txt” using a binary-search algorithm. First, the rockyou list is stored in a vector data structure. For the binary search, the rockyou vector must be sorted, which can be done through a merge sort. Once the vector is stored and sorted, it can be used an indefinite number of times, depending on how many passwords are tested. This entire procedure is O(nlog(n) + klog(n)), where n is the length of the rockyou list and k is the number of passwords tested.

2.2 Brute-Force Examination

While the wordlist examination is relatively straightforward, there are several possible approaches for a brute-force test. A simple approach would be to use a regular expression of a pre-determined length, but that would not be rigorous enough. As such, this project has evaluated a few other approaches that would provide a more comprehensive understanding of how safe the password is.

The first approach taken tests a brute-force algorithm in real-time by checking every possible combination of ASCII characters through recursion. Every string possibility would be created and subsequently compared with the password being tested. Using chrono, a timer would also be active to determine how much time passes before the password is found. If the password is not found within the allotted timeframe, then the password is considered safe. The problem with this approach is the significant overhead. The algorithm to create these strings is O(nk), where n is the number of ASCII characters tested and k is the maximum length of passwords tested.

To avoid some of the problems with recursion, another approach would be to construct a wordlist with the string combinations beforehand, then use a linear search to find the password. This allows for more control with what words are checked, as the wordlist can be configured beforehand.

2.3 Social Engineering Examination

Social engineering is the act of guessing security flaws and passwords based on known human behavior. While social engineering was not tested in the scope of this project, there were a few theories that could be expanded on in later endeavors.

* Does the password contain any personal information relating to the user, their family, or their friends?
* Is the password some variation of the username?
* Is the password related to its corresponding system?
  + (Ex: A password for Gmail being “gmail”)
* Is the password connected to current events?
* Is the password connected to popular culture?
* Is the user reusing an already leaked password?
* Can the password be found in a common dictionary?

In future endeavors, it would be best to examine each of these potentially dangerous avenues and make sure that users are not following these common mistakes.

3.1 Conclusion and Future Endeavors

In later prospects, the scope of the program should be expanded to include more wordlists, have less overhead, and utilize common social engineering flaws. In addition, the project should provide suggestions on safe passwords that the users could use as an alternative.

The procedure for making passwords is one that users need to take seriously. Bank accounts, classified documents, personal information, and a myriad of other sensitive information is at stake. If a password cannot survive past these simple attack methods, it could end up creating several disasters on a worldwide scale.

REFERENCES

[1] Nik Cubrilovic . 2009. RockYou Hack: From Bad to Worse. (December 2009). Retrieved July 22, 2021 from https://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook/passwords/

Conference Name:ACM Woodstock conference

Conference Short Name:WOODSTOCK’18

Conference Location:El Paso, Texas USA

ISBN:978-1-4503-0000-0/18/06

Year:2018

Date:June

Copyright Year:2018

Copyright Statement:rightsretained

DOI:10.1145/1234567890

RRH: F. Surname et al.

Price:$15.00