Project Proposal

Distributed Hash Cracking

CPE 512

KYLE RAY

September 28, 2017

Contents

Introduction 1

Project Summary 2

Test Layout 2

Environment 2

Goals 2

Introduction

In this day and age most of our daily life is performed in the digital realm. To protect our digital identity, which can grant access to everything from our bank accounts to our social security number, we must have a secure method to access this information. Passwords have been used since the earliest days of computing and there have been many techniques developed to keep these passwords secure. In this project, I would like to explore a form of securing passwords known as hashing and one of the attacks used to crack it.

Hashing is a one-way mathematical transformation of an input to a constant sized output string. Hence a user gives a password, which then passes through a "hashing" algorithm and is eventually stored. There are many different hashing algorithms out there which range in complexity. Some of the most common hashing techniques are MD5, SHA-1, SHA-256, and so on.

Example MD5 Hashed Password:

Password = "Parallel_Programming_1s_fUn"

Hash = 6243ebf30a0642f7dad0ec17cc916df9

The title of this proposal is "Distributed Hash Cracking" which implies that I want to explore the benefits of parallelizing a known hash cracking technique, the dictionary attack. This attack will take a list, usually very large list, of different known passwords, apply the hashing algorithm, and check this produced hash to the hash under question, the user's password. This can be quite a process intensive task if the dictionary is large and the utilization of a distributed cluster of computers can be utilized to break these tasks down into smaller subtasks and thus decrease the overall time it takes to successfully find the password.

Project Summary

In this project, I would like to utilize the Open Message Passing Interface (MPI) library for C++ to analyze the possible speedups of distributed hash cracking via the dictionary attack. The hashing algorithms that will be tested are the MD5, SHA-1, and SHA-256 algorithms.

Test Layout

The dictionary to be used in the tests can be formulated for the test or it is possible to use a list from a well-known source such as SkullSecurity. The user password will have to exist in the dictionary otherwise the test will not be very interesting. I would like to test each individual hashing algorithm with the same user password, to see the timing differences in overhead associated with each algorithm. I would also like to measure the distribution time based on dictionary size. The overall test should be looked at as if from the Hacker's point of view, where we don't really know that the password is in the list and we could be applying random dictionaries that we find on the web until we crack it. The pseudo test plan is as follows:

1.) Get number of MPI processes to be used from command line.
2.) Distribute a subsection of the dictionary provided as well as the user hash in question to each process.
3.) For each hashing algorithm (MD5, SHA-1, and SHA-256) Note: the following will be executed in parallel via the MPI processes.
    a. Gather a password from the local list.
    b. Apply the current hashing algorithm.
    c. Check against the user hash.
    d. If this is the password, report and stop operation for this hash algorithm.
4.) Display password.
5.) Wrap Up Timing Analysis

Environment

This test is to be carried out on either the Alabama Supercomputer or the Jetson cluster located at the University of Alabama in Huntsville.

Goals

The overall goals of this project are as follows:

1.) To become more familiar with parallel computing via the MPI interface.

2.) Gather timing and possible speedup information on a real-world problem that is mainly carried out on serial machines.
3.) See the effects produced by popular cryptographic algorithms that hinder and slow down hackers from gaining access to our systems and information.