**GRS CS 655: GENI Mini Project Final Report:**

**Password Cracker**

*Team Members:*

*1) Rhythm Somaiya – U84158310*

*2) Aman Ahmed – U01171262*

*3) Nirbhay Malhotra –*

***GitHub link:***

***Project on GENI:***

**1. Introduction/Problem Statement**

Whether it is accessing our email account or shopping online to purchase the essentials, passwords are an integral part of our everyday lives. Passwords are among the most fundamental mechanisms for securing data. It is common today for password-protected systems to store passwords in their databases in the form of hashes. The MD5 hashing algorithm generates digests that are 128 bits in length. However, even though vulnerabilities have been discovered in the algorithm's security, it continues to be widely used on the Internet.

According to the project title, this project will address the "Password Cracker" problem, where we intend to decrypt passwords based on their MD5 hash. Given such an MD5 hash made up of 5 alphabetic characters, the objective is to be able to figure out what the original password that was hashed was.

Since MD5 works only in one direction, it is impossible to decrypt a hashed password. This can only be accomplished by brute-forcing every possible 5-character alphabetic password to find its MD5 hash and then comparing it with the submitted hash. Since brute-force algorithms can take a considerable amount of time, one of the most efficient ways to accomplish this would be to use a distributed or multi-threaded approach. In the case of uppercase and lowercase characters, each of the five characters in the input has 52 possible combinations, which means that the system can iterate through 525 = 380204032 possible password combinations. Regarding the architecture of this project, we intend to develop a method for cracking an MD5 password hash by utilizing a distributed system in GENI. There will be a central server and a web interface where the MD5 hash will be passed. The hash will be based on a five-character password consisting only of alphabets (either lowercase or uppercase) for the purpose of this project. This message digest is input by the user via a web interface, and once it has been processed on the back end by the nodes in a distributed manner, the original password should be returned. The objective of this assignment is to determine how to configure the connections between a web page, a server node, and a set of multiple worker nodes so that they will be able to communicate together to crack the MD5 hash. The worker nodes must be able to cooperate so that they can try different passwords and see if they match the input, as well as stop their processes once one of them has completed the process. It is also necessary for the web page and the server node to communicate to receive the input and return the cracked password. The server node must also communicate with the worker nodes to properly distribute work among them.

**4. Conclusion**

Overall, the main findings of this project illustrate the importance of distributing such a workload. With one node, we would start iterating at the letter “A” but with multiple worker nodes, we can start with the next alphabet in an alternating way thus, not having too much of a concern about passwords starting with letters later in the alphabet. In terms of scalability and workload distribution, it was the architecture of the network and how the jobs were distributed that had a greater effect than the TCP network conditions. In its current state, the project does not allow the user to actually change the number of workers on the fly based on what is specified. However, as a future extension, the project could be extended to enable the number of worker nodes to be set up on the spot for each user. Additionally, this project could be extended to include longer passwords, alphanumeric passwords, allowing the user to specify parameters such as bandwidth and delay, caching previously found MD5 hashes to prevent re-compiling them in future rounds or iterations, and others.

**5. Division of Labor**

- Aman and Rhythm focused on the frontend implementation using React and worked on the backend using a Java framework – Spring to make a RESTful web service and handle the API calling procedure.

- Nirbhay worked on the backend architecture part of this project with all three of us contributing on the testing and other implementation tasks required.

**REFERENCES**

[1] GENI Tutorial Repository, “<https://groups.geni.net/geni/wiki/GENIExperimenter/Tutorials>”

[2] GENI-Based Classroom Exercises, “<http://www.cs.unc.edu/Research/geni/geniExercises/>”

[3] MD5,”<https://en.wikipedia.org/wiki/MD5>”

[4] ReactJS, “<https://reactjs.org/tutorial/tutorial.html>”

[5] Java Spring, “<https://spring.io/guides>”