Password Cracker

Aishwarya Reddy Lachangar(U62138442), Anoohya Veerapaneni(U43071771)

Github link: CS655-FinalProject

Project on GENI:

- Name of GENI Slice: fp2
- Project: CS655-Fall2022

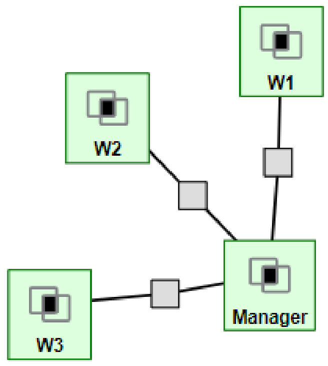
1. Problem Statement

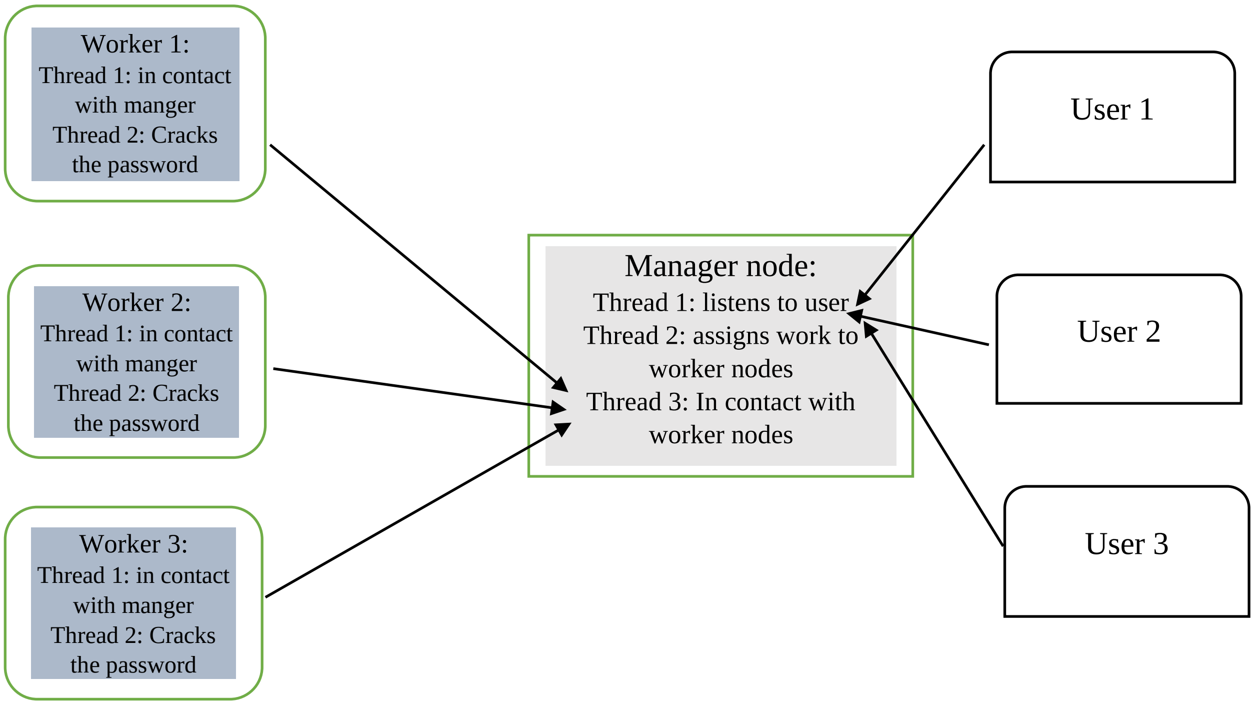
1. Definition:
   In this project we designed and created a distributed system that can crack passwords which is encrypted by md5. Users will submit this md5 hash password or any 5 characters to the system through the web interface we developed. The system will parse all the requests with the management service. So, the management device will allocate the cracking jobs to the workers and scales on its own, whenever it gets new requests. The workers will use the brute force approach to crack the passwords.

2. Motivation/Learning Outcomes:
   By implementing this project, we were able to learn
   - socket programming
   - web front-end nginx
   - multi threads programming
   - md5 hashing
   - GENI testing
   - distributions in system
   - Brute force to crack password

2. Design/Setup

1. Setup Diagram: Below is the screenshot from the GENI portal which shows manager and workers.




Worker 1:
Thread 1: in contact with manger
Thread 2: Cracks the password
Worker 2:
Thread 1: in contact with manger
Thread 2: Cracks the password
Worker 3:
Thread 1: in contact with manger
Thread 2: Cracks the password
Manager node:
Thread 1: listens to user
Thread 2: assigns work to worker nodes
Thread 3: In contact with worker nodes
User 1
User 2
User 3

2. Environment:

The environment of the experiment is as following:

Linux version	Linux version 4.15.0-121-generic
Java version	Jdk 11.0.9.1/jre 11.0.9.1
Nginx	Nginx/1/14/0(Ubuntu)

The assumption of our system is as following:

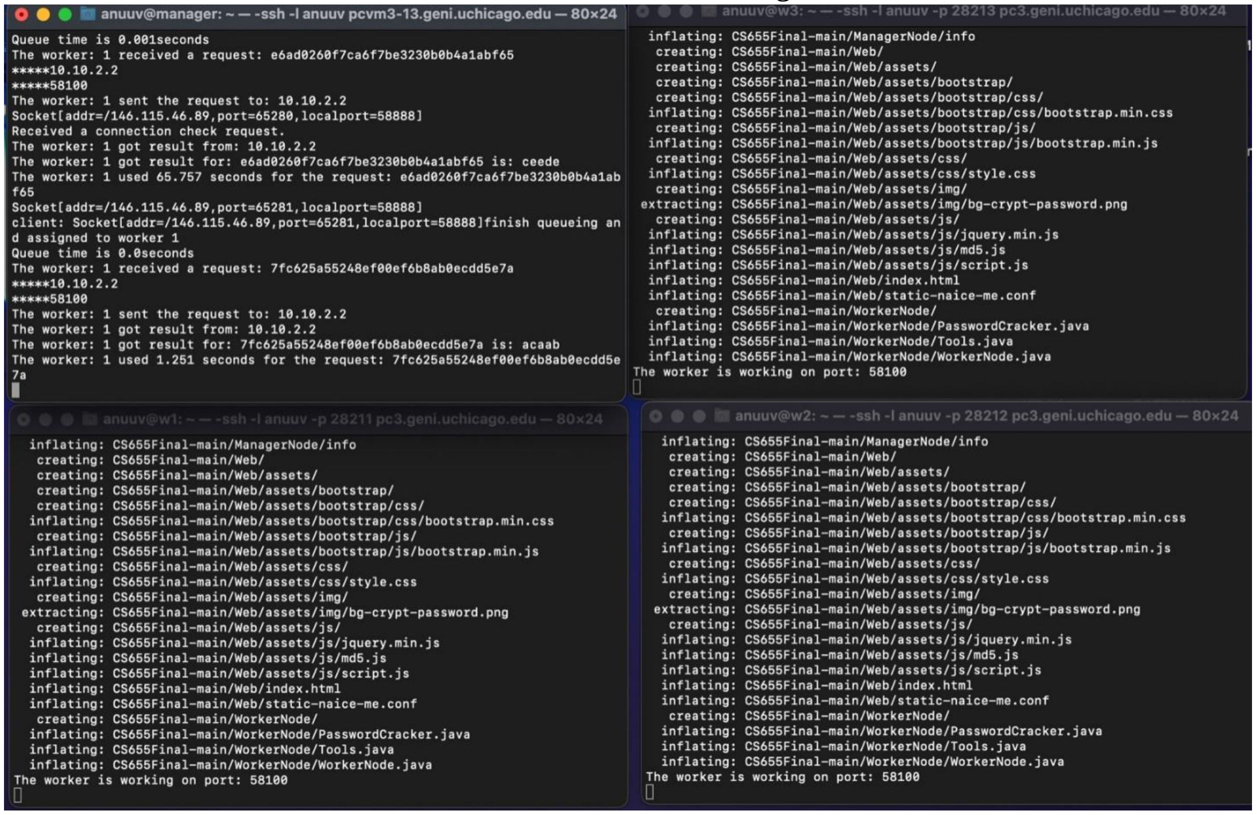
Maximum number of requests that can handle in parallel	3
Maximum number of workers	Greater than 3

Based on the assumptions, we designed experiments to evaluate the performance of the system. We would test in turn:

● When the number of users is fixed, the influence of different numbers of workers on the system transmission time and password calculation time.
● When the number of workers is fixed, the influence of different users on the system transmission time and password calculation time.
● The influence of password cracking difficulty on the transmission time and cracking time.

3. Execution

1. Configuration: same as Reproducibility
   Here are the screenshot, how it looks after successful running of the commands:


anuuv@manager: ~ — -ssh -l anuuv pcvm3-13.geni.uchicago.edu — 80×24
anuuv@w3: ~ — -ssh -l anuuv -p 28213 pc3.geni.uchicago.edu — 80×24
anuuv@w1: ~ — -ssh -l anuuv -p 28211 pc3.geni.uchicago.edu — 80×24
anuuv@w2: ~ — -ssh -l anuuv -p 28212 pc3.geni.uchicago.edu — 80×24

```
Queue time is 0.001seconds
The worker: 1 received a request: e6ad0260f7ca6f7be3230b0b4a1abf65
*****10.10.2.2
*****58100
The worker: 1 sent the request to: 10.10.2.2
Socket[addr=/146.115.46.89,port=65280,localport=58888]
Received a connection check request.
The worker: 1 got result from: 10.10.2.2
The worker: 1 got result for: e6ad0260f7ca6f7be3230b0b4a1abf65 is: ceede
The worker: 1 used 65.757 seconds for the request: e6ad0260f7ca6f7be3230b0b4a1ab
f65
Socket[addr=/146.115.46.89,port=65281,localport=58888]
client: Socket[addr=/146.115.46.89,port=65281,localport=58888]finish queueing an
d assigned to worker 1
Queue time is 0.0seconds
The worker: 1 received a request: 7fc625a55248ef00ef6b8ab0ecdd5e7a
*****10.10.2.2
*****58100
The worker: 1 sent the request to: 10.10.2.2
The worker: 1 got result from: 10.10.2.2
The worker: 1 got result for: 7fc625a55248ef00ef6b8ab0ecdd5e7a is: acaab
The worker: 1 used 1.251 seconds for the request: 7fc625a55248ef00ef6b8ab0ecdd5e
7a
```

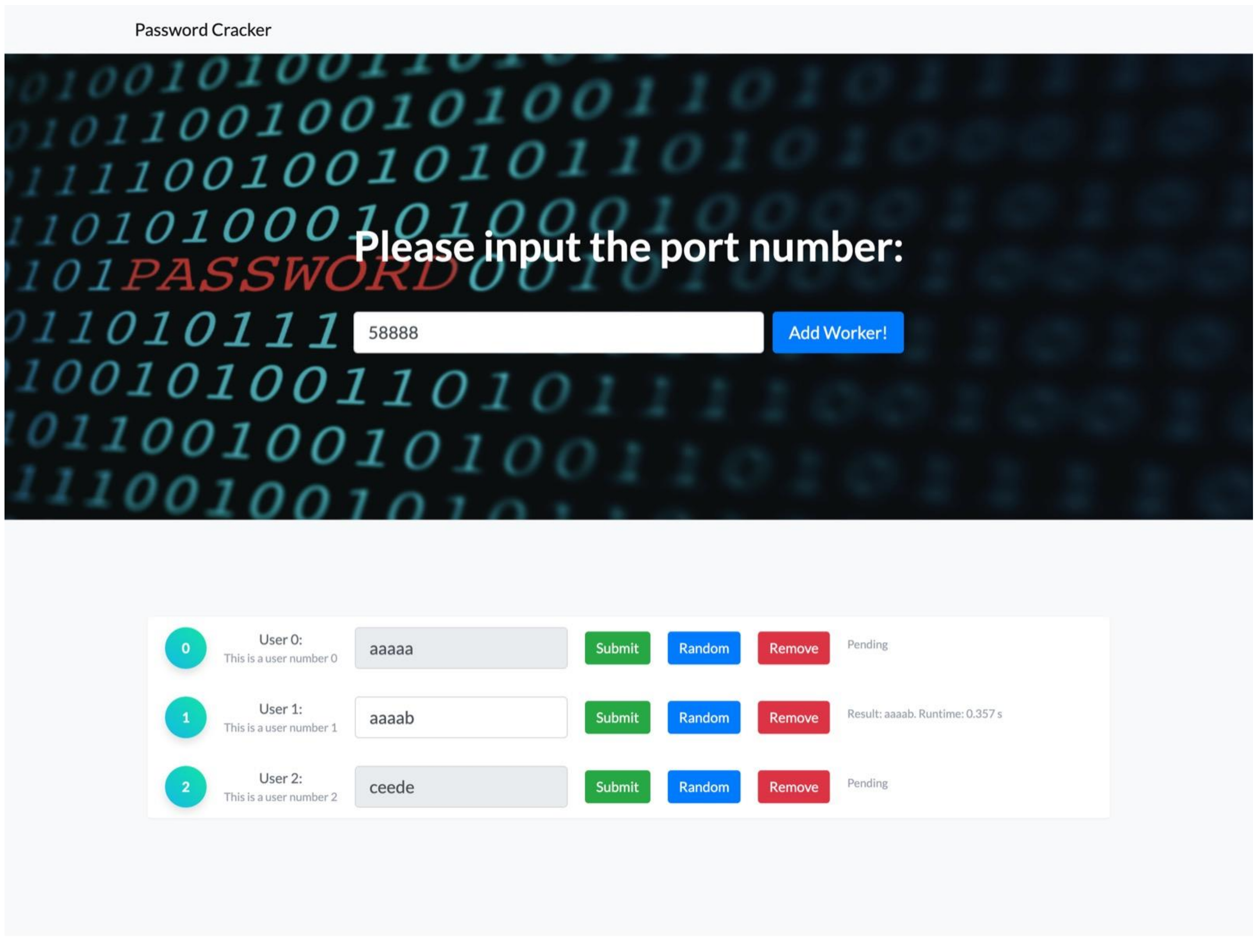
```
  inflating: CS655Final-main/ManagerNode/info
   creating: CS655Final-main/Web/
   creating: CS655Final-main/Web/assets/
   creating: CS655Final-main/Web/assets/bootstrap/
   creating: CS655Final-main/Web/assets/bootstrap/css/
  inflating: CS655Final-main/Web/assets/bootstrap/css/bootstrap.min.css
   creating: CS655Final-main/Web/assets/bootstrap/js/
  inflating: CS655Final-main/Web/assets/bootstrap/js/bootstrap.min.js
   creating: CS655Final-main/Web/assets/css/
  inflating: CS655Final-main/Web/assets/css/style.css
   creating: CS655Final-main/Web/assets/img/
 extracting: CS655Final-main/Web/assets/img/bg-crypt-password.png
   creating: CS655Final-main/Web/assets/js/
  inflating: CS655Final-main/Web/assets/js/jquery.min.js
  inflating: CS655Final-main/Web/assets/js/md5.js
  inflating: CS655Final-main/Web/assets/js/script.js
  inflating: CS655Final-main/Web/index.html
  inflating: CS655Final-main/Web/static-naice-me.conf
   creating: CS655Final-main/WorkerNode/
  inflating: CS655Final-main/WorkerNode/PasswordCracker.java
  inflating: CS655Final-main/WorkerNode/Tools.java
  inflating: CS655Final-main/WorkerNode/WorkerNode.java
The worker is working on port: 58100
```

```
  inflating: CS655Final-main/ManagerNode/info
   creating: CS655Final-main/Web/
   creating: CS655Final-main/Web/assets/
   creating: CS655Final-main/Web/assets/bootstrap/
   creating: CS655Final-main/Web/assets/bootstrap/css/
  inflating: CS655Final-main/Web/assets/bootstrap/css/bootstrap.min.css
   creating: CS655Final-main/Web/assets/bootstrap/js/
  inflating: CS655Final-main/Web/assets/bootstrap/js/bootstrap.min.js
   creating: CS655Final-main/Web/assets/css/
  inflating: CS655Final-main/Web/assets/css/style.css
   creating: CS655Final-main/Web/assets/img/
 extracting: CS655Final-main/Web/assets/img/bg-crypt-password.png
   creating: CS655Final-main/Web/assets/js/
  inflating: CS655Final-main/Web/assets/js/jquery.min.js
  inflating: CS655Final-main/Web/assets/js/md5.js
  inflating: CS655Final-main/Web/assets/js/script.js
  inflating: CS655Final-main/Web/index.html
  inflating: CS655Final-main/Web/static-naice-me.conf
   creating: CS655Final-main/WorkerNode/
  inflating: CS655Final-main/WorkerNode/PasswordCracker.java
  inflating: CS655Final-main/WorkerNode/Tools.java
  inflating: CS655Final-main/WorkerNode/WorkerNode.java
The worker is working on port: 58100
```

```
  inflating: CS655Final-main/ManagerNode/info
   creating: CS655Final-main/Web/
   creating: CS655Final-main/Web/assets/
   creating: CS655Final-main/Web/assets/bootstrap/
   creating: CS655Final-main/Web/assets/bootstrap/css/
  inflating: CS655Final-main/Web/assets/bootstrap/css/bootstrap.min.css
   creating: CS655Final-main/Web/assets/bootstrap/js/
  inflating: CS655Final-main/Web/assets/bootstrap/js/bootstrap.min.js
   creating: CS655Final-main/Web/assets/css/
  inflating: CS655Final-main/Web/assets/css/style.css
   creating: CS655Final-main/Web/assets/img/
 extracting: CS655Final-main/Web/assets/img/bg-crypt-password.png
   creating: CS655Final-main/Web/assets/js/
  inflating: CS655Final-main/Web/assets/js/jquery.min.js
  inflating: CS655Final-main/Web/assets/js/md5.js
  inflating: CS655Final-main/Web/assets/js/script.js
  inflating: CS655Final-main/Web/index.html
  inflating: CS655Final-main/Web/static-naice-me.conf
   creating: CS655Final-main/WorkerNode/
  inflating: CS655Final-main/WorkerNode/PasswordCracker.java
  inflating: CS655Final-main/WorkerNode/Tools.java
  inflating: CS655Final-main/WorkerNode/WorkerNode.java
The worker is working on port: 58100
```

   Here the one clicked first will be assigned to Worker 1, then later to other workers

CS655 FINAL MINI PROJECT

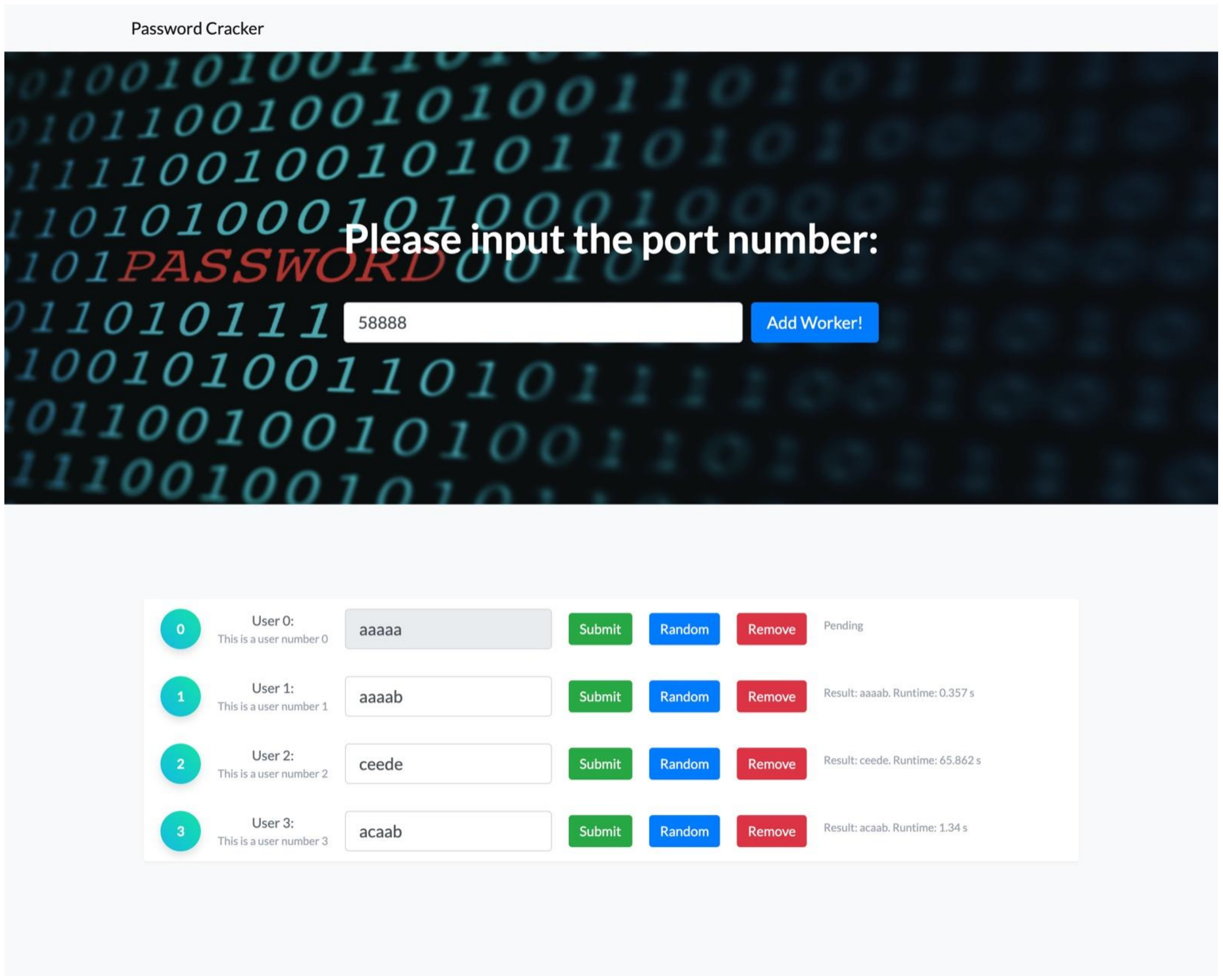


58888

User 0: This is a user number 0 — aaaaa — Pending

User 1: This is a user number 1 — aaaab — Result: aaaab. Runtime: 0.357 s

User 2: This is a user number 2 — ceede — Pending



58888

0	User 0: This is a user number 0	aaaaa	Pending
1	User 1: This is a user number 1	aaaab	Result: aaaab. Runtime: 0.357 s
2	User 2: This is a user number 2	ceede	Result: ceede. Runtime: 65.862 s
3	User 3: This is a user number 3	acaab	Result: acaab. Runtime: 1.34 s

2. Analysis:

According to the experiments, it is obvious that generally, the average cracking time and the average transmission time remain unchanged with the change of the number of workers and the number of requests, while the average queueing time becomes larger when the number of workers decreases and the number of requests increases. Based on the results, we could generally conclude that the total running time of the system is mainly determined by the queueing time. Since we have a limitation of total number of workers, if all worker nodes are occupied, the remaining cracking requests should wait in the queue until one of the workers finishes its job, and therefore the curve has an inflection point when the number of workers reaches the upper limitation. Besides, as the difficulty of password cracking increases, the total time also increases linearly.

4. Demo:

Added in the github link

5. Reproducibility

1. Scripts to setup:

ManagerNoder.sh & WorkerNode.sh

2. Code & Code quality:
   Added commits for every possible code and also pushed into git with comments as much as possible, code readability is very easy. Explained everything how to in the steps.
3. Rspec files: it is in github
4. Instruction to reproduce experiment:

STEPS:

1. Setup a GENI slice and upload Rspec.text given above into the add resources.
2. Then Reserve the resources, if you want to change the server, then you need to update the following into the code:
   a. `sudo echo ' server_name pcvm3-13.geni.uchicago.edu;'` to the specific resources you have used.
   b. In the info under ManagerNode package which is as

      3(number of worker)
      58100
      10.10.1.2 10.10.3.2 (the first IP address on manger node)
      10.10.2.1 10.10.2.2 & second IP address is related worker node)
      10.10.3.1 10.10.1.2

3. Next step would be to wait open terminal for each node(manager and 3 workers)
4. And first run the bash scripts as
   a. ManagerNode –
      `wget https://github.com/anoohya29/CS655Final/blob/main/BashScripts/ManagerNode.sh`
   b. WorkerNode –
      `wget https://github.com/anoohya29/CS655Final/blob/main/BashScripts/WorkerNode.sh`
5. After this type- cat ManagerNode.sh and cat WorkerNode.sh
6. Then run- sudo bash WorkerNode.sh on all the worker terminals, wait till you get "The worker is working on port 58100" on all the worker terminals.
   Then run- sudo bash ManagerNode.sh on manager node, wait till you get "The manager is working on port- 58888"
7. Now click on this link- " http://pcvm3-13.geni.uchicago.edu"
8. Now you will the below page on your browser:

Conclusion:

Summary:

Overall, with this project, we were able to implement the web interface through which the passwords are sent. And the management device which will distribute the jobs to the worker nodes deployed on different machines. This system is available for the users to submit their cracking requests through the webpage. So, his request will be sent to the manager node (management device) and then it dedicates the work to the different worker nodes by following the implemented strategy. Then it will again return to the server and show it on the web interface when the password is cracked. This result will the total time including the transmission time, queuing time, and cracking time. The total time increases with the increase in number of users and passwords and the difficulty level of it, also if the number of worker nodes decreases.

Possible extensions:

In future, we can add more nodes to the system, so that we can decrease the amount of time. We can create worker nodes that can be in different sites on the GENI server. This way the passwords could be cracked with more speed. We can also try the different queuing techniques and different strategies to be able to allow the worker nodes do the work in different procedure. It can also improve the efficiency of the system.

Work division

Design the logical structure of the system, writes the multi-thread tasks on the manager and the worker nodes, performs multiple tests on the system, configures the environment on the GENI	Aishwarya Reddy Lachangar (U62138442)
Design the logical structure of the system, constructs the web interface of the system, deploys the web server on the corresponding nodes, create the executable shell file	Anoohya Veerapaneni (U43071771)
Report	Aishwarya Reddy Lachangar and Anoohya Veerapaneni