CS 2302 Data Structures

Lab 1 Report

Carlos Montgomery

80577684

I chose Lab Option C, and this required for the use of recursion to find the actual passwords of each of the hundred users. The goal of the lab was to create a randomly generated password, after we have to create a new string with the salt value and hash the newly created password. After so we had to compare and search if that password matched any of the already provided passwords.

I first thought that I could crack all the codes was to go through all the possible combinations on integers from within the given range and create the new string. So, to do I start off with 000 and went up by one and concatenate with the salt value. I first had to read the file and store each of the variables to be able to compare later on. I used a for loop to open the file and I created three lists that each had assigned to their specific value.

Once the values were stored, the password generation would take place. Then the creation of a new string with the salt value and the random number was stored and hashed with the already provided hash function. Once the hash password was returned I used and if statement that would compare the newly hashed password to the hashed passwords from the text file that was provided.

For my test I created new hashed passwords with different hash values and began to compare them, I also shortened the text file that was provided. The password length was also altered to get different passwords.

I was later informed that I had not done what the lab required, I seemed to not understand the question of the lab. So, for later reference I plan on asking and better understanding of what is required. To my understanding I had to create a random integer recursively and then concatenate it then compare, but I thought of comparing every combination to be able to get every password found. So, to conclude I did not do the lab correctly I was doing a trivial task and which at the end of the day took for every to “crack” the passwords.

Appendix

|  |
| --- |
| import hashlib |
|  | import string |
|  | import itertools |
|  |  |
|  |  |
|  | def hash\_with\_sha256(str): |
|  | hash\_object = hashlib.sha256(str.encode('utf-8')) |
|  | hex\_dig = hash\_object.hexdigest() |
|  | return hex\_dig |
|  |  |
|  |  |
|  | def main(): |
|  | hex\_dig = hash\_with\_sha256('This is how you hash a string with sha256') |
|  | print(hex\_dig) |
|  |  |
|  |  |
|  | # This reads the "password\_file.txt" and stores each aspect of the record to it's own variable |
|  | def read\_file(): |
|  | file\_open = open("password\_file.txt", "r") |
|  | for line in file\_open: |
|  | fields = line.split(",") # This used to separate and read in between the comas |
|  | # Each field to the record is stored to their own variable |
|  | username = fields[0] |
|  | salt\_value = fields[1] |
|  | hash\_password = fields[2] |
|  | # print("Username: " + username + " Salt Value: " + salt\_value + " Password: " + hash\_password) |
|  | # this is used to test if each value is correctly stored |
|  | return salt\_value |
|  |  |
|  |  |
|  | # The following code is used to create every possible combination of digits. |
|  | digits = string.digits |
|  | def method2(min,max): |
|  | num = itertools.product(digits, repeat=min) |
|  | for prod in itertools.product(num): |
|  | string = ''.join([''.join(k) for k in prod]) |
|  | file\_open = open("password\_file.txt", "r") |
|  | for line in file\_open: |
|  | fields = line.split(",") # This used to separate and read in between the comas |
|  | # Each field to the record is stored to their own variable |
|  | username = fields[0] |
|  | salt\_value = fields[1] |
|  | hash\_password = fields[2] |
|  | new\_password = salt\_value+ str(string) |
|  | if (hash\_with\_sha256(new\_password) == hash\_password): |
|  | print(new\_password + " "+ hash\_password ) |
|  | else: |
|  | print(hash\_with\_sha256(new\_password)+ ' = '+ hash\_password) |
|  | if min < max: |
|  | return method2(min+1,max) |
|  |  |
|  |  |
|  | method2(3,7) |

I, **Carlos Montgomery,** certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class.