ASSIGNMENT 1.2

-NIKHIL REDDY
N19337907

ASHLEY MADISION

Looking at the password dump file, I noticed that this was a file which followed a pattern of data storage without any encryption or hashing. The file consisted of Comma Separated Values(CSV) entries. I used the basic Linux command "awk" to separate the password from the each query line as the third entry of every line was the password in this case.

```
awk '{print $1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$13,
$4;}' AshleyMadison.txt > AshleyMadisonCracked.txt
```

The above command prints all the existing 13 fields in each query followed by the password into a new file called AshleyMadisonCracked.txt

EHARMONY

Looking at the eHarmony password dump file, I noticed that they were all of the same length 32 characters. I figured that they could not have been encrypted passwords as encryption does not result in a fixed length cipher text. Thus, I realized it was a Hash list and with the help of a little research on the internet, I was able to figure out that this was an MD5 hashing technique. To check if the method was really MD5, I used online tools like hashkiller, etc. to check with their MD5 hash database and I found matches.

Now since this was a Hashing method, I could not "de-hash" or decrypt it. Figuring out the passwords would have to be achieved by matching hashes of known plaintext and the given hashes. I thought of downloading or making my own rainbow tables but due to computation and processing time constraints, I shortened my dictionary to 1100 commonly used passwords and their corresponding MD5 hashes. I then used BruteForce method to check if there were any matching hashes. But unfortunately, the dictionary I had chosen was not large enough for the given set of password dump. I have written a generic code that inputs dictionary and checks with the password dump, so changing the input dictionary might actually help crack the passwords.

FORMSPRING

After figuring out the hashing technique used in the previous dump, it actually took lesser time to figure out that this particular password dump was also hashed but using SHA256 as this was almost the only popular 256 bit hashing algorithm and the announcement mail had confirmed it by then. But I realized that this was a salted SHA256 algorithm after finding out that there were no matches to SHA256 hash databases on online tools but found matches for salted SHA256 hash methods.
For cracking this password dump, I decided to use the same 1100 words dictionary of commonly used passwords. But this time there was a problem that these passwords were salted randomly with random numbers between 00 and 99. Luckily, since it was given that the salt was appended to the start of the password, I decided to use BruteForce Method to salt every word in the dictionary with all the numbers between 00 and 99 and then hash it using SHA256 and check for any hits in the password dump. Due to the larger dataset this time, I had shifted to Python as it handles larger data easily unlike JAVA. Unfortunately, this method also did not yield any hits for the given password dump.

CONCLUSION

In both MD5 and SHA256 password hit algorithms, I checked and confirmed my code using custom data input sets and they worked perfectly fine proving that increasing the dictionary size or perhaps using a more statistical analyzed word list might help generate more hits for these two password dumps.

BIBLIOGRAPHY

http://www.thegeekstuff.com/2010/01/awk-introduction-tutorial-7-awk-print-examples

http://www.passwordrandom.com/most-popular-passwords

http://arstechnica.com/security/2015/08/cracking-all-hacked-ashley-madison-passwords-could-take-a-lifetime/

http://qz.com/501073/the-top-100-passwords-on-ashley-madison/

http://stackoverflow.com/