Virginia Tech ■ ECE 5480 ■ Fall 2018

Project 2

Before starting this first homework assignment, please be sure that you have completed all of the following activities.

☐ · Review the Graduate Honor System at https://graduateschool.vt.edu/academics/expectations/graduate-honor-system.html. Review the Graduate Honor System Constitution, especially Articles I (Sections 1, 2, and 3), V, VI, VII, VIII, and IX.

Please note the following.

- Solutions must be clear and presented in the order assigned. Solutions must show work needed, as appropriate, to derive your answers. Written answers should be concise, but sufficiently complete to answer the question. Neat hand drawings, where needed, are acceptable. Your final solution for each problem must be easily identified.

- At the top of the first page, include: your name (as recorded by the university); your email address; and the assignment name (e.g., "ECE 5480, Project 2"). Do not include your Virginia Tech ID number or your social security number.

- Homework must be submitted as a PDF (.pdf) file with the file name *lastname_firstname*_PROJ2.pdf, where *lastname* is your last or family name and *firstname* is your first or given name. Note that additional files to be submitted are required for this project.

- Submit your assignment using the Assignments area of the class website.

INTRODUCTION

From your research, you've found that many IoT systems are vulnerable from poor password authentication. Rather than storing passwords in plaintext on the device, one common approach is to hash the password using a cryptographic hash function, and then store the hash values. This way, if an attacker is able to recover the hashed password file, it would be difficult for the attacker to recover the password from the hashed values.

We've talked about brute force and dictionary password attacks. Another type of attack is called a rainbow table attack, wherein the attacker uses a pre-computed table of hash values from common plaintext passwords. The idea is that rather than having to pick a plaintext password from a list, compute the hash value, then compare that hash value to a value from the hashed password list, the attacker just has to pick a hashed value from the password list and see if it appears in the rainbow table. If there is a match, the rainbow table gives the corresponding plaintext password.

For this project using Python you will explore these concepts to create a simple rainbow table from a list of common plaintext passwords. You will then try to find the plaintext passwords from a list of hashed passwords that you, the attacker, have recovered.

DESIGN SPECIFICATION

The file containing the list of recovered password hashes is:
RECOVERED_PASSWORD_HASHES.txt.

The file containing the list of common plaintext passwords is:
10K_PLAINTEXT_PASSWORDS.txt.

From your reconnaissance of the IOT system, you know that the IOT device hashes user plaintext passwords one time using MD5.

Modify the Python script create_rainbow_table_INCOMPLETE.py to read in each entry of 10K_PLAINTEXT_PASSWORDS.txt, hash each plaintext password with MD5, and write each hashed password to the file RAINBOW_TABLE.txt. Your main task here is to hash the passwords and write the hashes to a text file.

Modify the Python script recover_hashed_passwords_INCOMPLETE.py to read a recovered hashed password, see if that hashed password exists in the rainbow table, then find the corresponding plaintext password; output the matches if they are found and record

how long it takes to search for each password. Your main task here is to put the timing commands in the correct places.

SUBMISSION

Your pdf report should contain for each captured password hash value:

- #: password hash value
- text: recovered plaintext password (NA if you were not able to recover the password)
- #: time (s) to search the entire rainbow table (if hash is not located) OR time to recover plaintext password (if hash is found). See TIMING EXAMPLE below.

Finally include answers to these questions:

1. Min time to recover a password
2. Maximum time to successfully recover a password
3. Time needed to search the entire rainbow table when no password could be recovered.

When submitting your solutions, include in your report the output of your recover_passwords.py program and the answers to the questions I asked above. Include with your report as separate files the source code for both .py programs you edited as well as the RAINBOW_TABLE.txt.

TIMING EXAMPLE

In Python, as an example, for this project you could generate timing statistics using the following:

```
# best timer to use across windows/unix platforms

from timeit import default_timer as timer

startTime = timer()

# your code to search the hash list for a match
```

```
endTime = timer()

print ('It took ', (endTime - startTime)*1000000, ' microseconds.')
```

FILE I/O EXAMPLE

To read from or write to a file stored on the harddrive, you use the open() function to create a file object:

```
fp = open(filename, mode)
```

Where filename is the string name of the file, and mode depends on whether you want to read from a file (“r”) or write to a file (“w”).

Here is an example of opening a file for reading:

```
fp1 = open("10K_PLAINTEXT_PASSWORDS.txt", “r”)
```

Here is an example of opening a file for writing:

```
fp2 = open("foo.txt", "w")
```

Once reading or writing a file, it is a best practice to close the file pointer:

```
fp.close()
```

Once the file has been opened for reading, one way to read in the lines of the text file into a list variable is:

```
lines = fp.readlines()
```

Since every line as the newline character at the end, a better way is to first strip off end control characters:

```
lines = [line.rstrip() for line in fp.readlines()]
```

To write lines of text to a file, you can use the write() function, though remember to append an ending newline character before writing the text to the file:

```
for line in lines:
```

```
        fp.write(line + "\n")
```