Part 1: Writing a Password Cracker

Objective
This is a small programming assignment to get the feeling for how a password cracking program works. You also should learn some about one-way encryption and why choosing a good password is important.

Background info

If you use C:

The Unix crypt(3c) function is used to encrypt a password. It is a function that implements a one-way encryption algorithm used for password encryption and requires #include < unistd.h>. It takes two const char* parameters and returns a char*. The first parameter is the password to encrypt (only the first 8 chars are significant). The second parameter is a "salt" used by the encryption algorithm. The salt is a two-character string chosen from the set [a-zA-Z0-9/]. On a successful encryption crypt() returns a pointer to the encrypted password. The string returned is 13 characters (from the set [a-zA-Z0-9/]) long and the first two chars are the salt. On an error it returns a null pointer. (Linux note: you need to link with libcrypt, -lcrypt). Be sure to read the entire man page Linux: man 3 crypt.

If you use python:

15.1. hashlib — Secure hashes and message digests

https://docs.python.org/3/library/hashlib.html

Source code: Lib/hashlib.py

This module implements a common interface to many different secure hash and message digest algorithms. Included are the FIPS secure hash algorithms SHA1, SHA224, SHA256, SHA384, and SHA512 (defined in FIPS 180-2) as well as RSA’s MD5 algorithm (defined in Internet RFC 1321). The terms “secure hash” and “message digest” are interchangeable. Older algorithms were called message digests. The modern term is secure hash.

Since crypt() or hashlib provide a one way encryption the password cannot simply be decrypted. So given an encrypted version of a password it seems that we cannot get the plaintext password. However, we can make guesses and see if one of our guesses encrypts to the same thing. If it is then chances are we have guessed the password (or at least something that encrypts to the same thing, which is all we really need). Knowing these things one can write a password cracker. A
brute force cracker would try all possible passwords. If we know properties of the password this could be feasible. Other crackers (hashcat, john the ripper) use rules and a dictionary of words to generate guesses (like capitalize the first char of a word and follow with a number).

**Assignment**

Now, finally to the assignment. You will write a password cracker that uses some rules and does a brute force attack on a set of passwords given those rules. Your program must be written in C++ or Python and work on any size password file that was encrypted using the MD5 encryption. The format of the password file is (with one entry per line):

```
username:encryption[:otherstuff]
```

[:otherstuff] is optional (some password files have it, some do not, so your program needs to be able to handle both ways).

**Rules to implement in your password cracker**

- All numbers (4 digits to 6 digits in length).
- A four char word from `/usr/share/dict/words` (Linux or Mac) which gets the first letter capitalized and a 1-digit number appended.
- A five char word from `/usr/share/dict/words` with the letter 'e' in it which gets replaced with the digit 3. (words with 2 2's treat as two separate words with e's, eg. sleep -> sl3ep and sle3p, but not sl33p).
- Any number of chars single word from `/usr/share/dict/words` (Linux or Mac)

Your cracker should work as follows. In a loop generate guesses based on the rule you are implementing. When you have found a match print it out to standard output and to a file in the format encrypted:password. Your program should stop once it has cracked all the passwords in the file or you run out of combinations for that rule. You may want to generate test cases of your own as well.

**Some samples to test your cracker is working**

```
1234 -> homer:81DC9BDB52D04DC20036DBD8313ED055:20:Homer Simpson:/home/homer:/bin/tcsh
Hair6 -> marge:2EB905F43B06291FF6BF9C951322300E4:351:20:Marge Simpson:/home/marge:/bin/tcsh
b3ach -> lisa:BCF784A318595866E15C89AC07DS2FE9:353:20:Lisa Simpson:/home/lisa:/bin/tcsh
9wXy! -> maggie:F0B93C4FF636C8B704A392771F914CB6
```

Note that your code will also be tested with other passwords.

**Part 2: Cryptography**

1. **DES**

*(weak DES keys)* There are four so called weak DES keys. One of those keys is $K = 00011111 00011111 00011111 00011111 00001110 00001110 00001110.

a) What happens if you use this key?

b) Can you find the other three weak keys?
2. RSA

In this exercise we consider an RSA modulus \( n = p \times q \) where \( p \) and \( q \) are large prime numbers (here, by large we mean at least equal to 5). We consider a valid RSA exponent \( e \) for RSA.

a. Show that neither \((p \mod 3)\) nor \((q \mod 3)\) can be equal to 0.
b. Under which condition \( e \) is a valid exponent for a modulus \( n \)?
c. From now on, we will assume that \( e = 3 \). Show that neither \( p - 1 \) nor \( q - 1 \) can be multiples of 3.
d. Deduce that \( p \mod 3 = q \mod 3 = 2 \).
e. What is the value of \( n \mod 3 \)?

3. Elliptic Curve

List the points on the elliptic curve \( E: y^2 = x^3 - 2 \pmod{7} \)

Draw the curve : https://www.desmos.com/calculator/ialhd71we3. What are the points \( P, Q, R \)?

Part 3: Blog Entry

Background

They say that one of the best ways to learn a foreign language is to immerse yourself in it. If you want to learn French, move to France. This assignment is designed to give you an opportunity to think about security during non-course related activities, such as when you're reading news articles, talking with friends about current events, or when you're reading the description of a new product on Slashdot. Thinking about security will no longer be a chore relegated to the time you spend in lecture, on assigned readings, on textbook assignments, or on labs. You may even start thinking about security while you're out walking your dog, eating breakfast, at the gym, or at a movie. In short, you will be developing "The Security Mindset" and will start thinking like a seasoned security professional.

It is also extremely important for a computer security practitioner (and actually all computer scientists) to be aware of the broader contextual issues surrounding technology. Technologies don't exist in isolation, rather they are but one small aspect of a larger ecosystem consisting of people, ethics, cultural differences, politics, law, and so on. This assignment and the use of the forum will give you an opportunity to discuss and explore these "bigger picture" issues as they relate to security.

Current Event Review

Current events reviews should be short, concise, very thoughtful, and well-written. Imagine a broad audience (a general technical audience). Your goal should be to write an article that will
help this audience learn about and understand the computer security field and how it fits into the broader context.

Your article should:

1. Summarize the current event;
2. Discuss why the current event arose - who did what, and why?
3. Reflect on what could have been done different prior to the event arising (to perhaps prevent, deter, or change the consequences of the event);
4. Describe the broader issues surrounding the current event (e.g., ethical issues, societal issues);
5. Propose possible reactions to the current event (e.g., how the public, policy makers, corporations, the media, or others should respond).