Computer Security
Coursework Exercice CW2

This coursework is a formative assessment. That means that we will be marking the submitted coursework, but that the mark will in no way contribute to your final mark for the course. The coursework is intended to help you learn more about computer security as well as help you understand how well you understand course material. We will be providing feedback by Friday 9th December. So if you do want to receive feedback you need to submit your solutions before that date.

1 Password Cracking

This question will involve (partially) cracking a list of hashed passwords. The password lists are based on the 2009 RockYou password leaks, which were randomly sampled.
Files `rockyou-samples.md5.txt`, `rockyou-samples.sha1salt.txt`, and `rockyou-samples.bcrypt.txt` store 100,000 password hashes each. These files can be found on dice in the `/group/teaching/cs` directory. Each function is progressively more resistant to password cracking than the previous one.

1.1 Brute-forcing MD5

The file `rockyou-samples.md5.txt` contains MD5 hashes of passwords, encoded in hexadecimal. Write a program in a programming language of your choice, which brute forces all five character passwords, using only numbers and lowercase ASCII letters (0-9 and a-z). The program should create an output file `md5-cracked.txt` which contains the passwords found and the number of occurrences for each of them. As the output format, keep one entry per line, with each entry being of the form ***n*,*password*** (E.g. `10,apples`). Finally, create a sorted output by running:

```
sort -t, -k1n,1 -o md5-cracked-sorted.txt md5-cracked.txt
```

Submit the file `md5-cracked-sorted.txt`.

Be aware that the naive implementation of brute forcing will not be sufficient here. You will need to check each possible password against all hashes very quickly, it is therefore strongly advised to use hashmaps (or, even better, multisets).

1.2 Cracking Common Passwords with Salted SHA-1

The file `rockyou-samples.sha1-salt.txt` contains SHA-1 hashes of passwords with a salt. The format of each line is `$SHA1p$`$salt$`$`$hash$, where:

$$hash = H_{\texttt{SHA-1}}(salt||password)$$

The approach to brute-forcing from Part 1 can't be used in this case, as there is no fast way to check a given password against the entire list. Instead, write a program which tries the 25 most common passwords against the entire list, and reports how often each occurred (in the same format at in part 1). The 25 most common passwords are:

123456	12345	123456789	password	iloveyou
princess	1234567	rockyou	12345678	abc123
nicole	daniel	babygirl	monkey	lovely
jessica	654321	michael	ashley	qwerty
111111	iloveu	000000	michelle	tigger

Sort your results in the same way as in Part 1, and save them as `salt-cracked-sorted.txt`. Submit this file.

1.3 Cracking bcrypt?

The file `rockyou-samples.bcrypt.txt` contains bcrypt hashes of passwords. Bcrypt is a more modern hash function designed for use with passwords. It's primary feature is that a bcrypt hash is (comparatively) difficult to compute, making brute force attacks far less effective than with the extremely efficient SHA-1 and MD5 hashes. The bcrypt hashes are stored in a standard format for bcrypt, and should be recognized by a bcrypt library of your choice. The hashes further automatically include a salt.

Write a program that finds the first five occurrences of the password `123456` by line number (counting from 1). Write each line number, in order, on a single line of the file `bcrypt-lines.txt`. Submit this file.

2 Spoofing email sender

For this exercise, you will send us an email with a spoofed email sender field:

- The subject line of your email should be your student id
- The sender of your email should be `mickey.mouse@disney.com`
- You will send your email to `cw2@vaniea.com`.

One way of doing this is by using the `mailx` utility program. You are free to try this amongst yourselves before you actually send your email to us.