Challenge 1: Users with (Weak) Passwords

Don't ask how, but we have obtained the SQLite databases of two different login systems. From what we have uncovered so far, it is likely that the users whose credentials are stored in the database are using weak passwords. I wonder who these users are?

Your challenge is to list all users and their **plaintext** passwords of the two different login systems. Of course, user passwords aren't just stored in plaintext in the databases. That would be too easy! The first login system implements Version 4 of the password storage methods discussed in class (i.e., storing usernames and hashed passwords with salt). The second login system implements Version 5 (i.e., using hash stretching). There may also be some rather annoying (albeit simplistic) obfuscation of the usernames in the database. Think ROT-*n*. And, yes, every username in a database is encrypted the same way.

You will be provided with SQLite databases for both backend databases; however, you will not have access to any login system frontend. You may, however, use any sample code discussed in class. You will also be provided with a dictionary of sample passwords to assist you in determining user passwords. User passwords are **guaranteed** to be in the dictionary. This seems too easy!

You are to write a program (or, likely, multiple programs) to complete the challenge. A **correctly** working program for Version 4 produces output that is structured as follows:

```
Version 4:
USERNAME > PASSWORD
jgourd > 5ph1nct3r
moneal > 123456789
kcherry > orangeiscool
[snip]
```

Of course, this assumes that you have correctly decrypted the usernames and obtained the plaintext passwords.

A **correctly** working program for Version 5 produces output that is structured as follows:

```
Version 5:
USERNAME > PASSWORD
jgourd > mercifulonthesecondthursdayoftheweek
moneal > tardinessisinexcusable
kcherry > supercalifragilisticexpialidocious
[snip]
```

I highly recommend that you use the sample PHP code provided in class as a starting point. In fact, passwords were hashed and stored in the database via PHP using functions similar to those discussed in class. However, you are free to use the language of your choice. You will want to manage your time and resources wisely, especially to ensure that you get code working quickly. On my pretty typical system, listing plaintext usernames and passwords for the Version 4 database using the provided dictionary took ~70 milliseconds. I know what you're thinking: "I should be done with this challenge in less than half an hour!" However, listing usernames and plaintext passwords for the Version 5 database using the provided dictionary took my system a little over 12 minutes!

You may want to pay attention to some specifics that may simplify things a bit for you:

- Version 4 implements a SHA-256 hash of a concatenation of a user's salt and password – **plus some fixed pepper**; that is, `sha256(salt + password + pepper)`. You will want to pay attention to the way in which the salt is stored in the database (hexadecimal, maybe?); moreover, the hash algorithm may expect it in another format (binary, maybe?)! Also, watch errant newlines on usernames, hashed passwords, and salt that is fetched from the database. By the way, the fixed pepper is a SHA-256 hash of the string "I really love this stuff!"
- Version 5 implements the bcrypt hash stretching algorithm – which just happens to be the default currently in use in PHP libraries.

When finished, upload a ZIP file containing the following items:

(1) A **PDF** document that provides: (a) the plaintext usernames and passwords for each login system version, and (b) thorough documentation of your process as you proceeded through the challenge; and
(2) All source code that you used to complete the challenge.

You will be graded on how well you:

(1) Work as a team to manage time and resources (5%);
(2) Programmatically fetch data from the login/registration system databases (2.5%);
(3) Test passwords from the provided dictionary against hashed passwords obtained from the database (2.5%); and
(4) Successfully list usernames and their plaintext passwords in the specified format (90%).

Each login/registration system is worth 45% of the total grade for this challenge (i.e., Version 4 is worth 45% and Version 5 is worth 45% – for a combined total of 90%). Users in the databases are each worth an equal number of points. For example, there are 37 users in the Version 4 database. Therefore, each successful listing of the username and plaintext password of a single user is worth approximately 1.22%. If this challenge is worth a total of 50 points, then each student in the Version 4 database is worth approximately 0.61 points. So, get all 37 plaintext passwords and earn all 22.5 points; get only 10 and earn 6.08 points.

As a hint, you will want to check if your code works on one or two database records before launching it on the entire database. This will be particularly helpful when working on Version 5.

GOOD LUCK!