CS 5460: Computer Security I
Fall 2019
Assignment 4: Secure Authentication
Total Marks: 175

Assume:
The authorities in your organization have recently decided to upgrade their authentication system through a three-step process:

- Developing a smart password meter/checker to help their customers with creating strong passwords that are not vulnerable to dictionary and targeted guessing attacks
- Leveraging cryptography hashing and salt to protect against offline guessing attacks
- Implementing lock-out rule to gain better protection against online guessing attacks

Now, considering your knowledge and expertise in cybersecurity, you have been assigned to this project that needs you and your team completing the following tasks as a part of prototype development. If your prototype works successfully, the organization would allocate full-scale budget to develop the complete version of secure authentication system.

A. Smart Password Meter / Checker [Marks: 110]

Task 1 [Marks: 40]
- Identify a list of 30 dictionary words commonly used to create passwords. **Submit** this list in a doc file.

- Users often create passwords using a variation of dictionary words, which include but are not limited to the following strategies:

- Password based on repeating a dictionary word twice or thrice (e.g., catcat, dogdogdog).
- Password based on a dictionary word spelled backwards (e.g., sttesuhcassam).
- Password based on quick adjustments, like making the first letter of a word capital, adding a digit as the second last or last character, and/or adding a common special character (e.g., !, @, #, $, %, &) at the end.
- Password based on replacing letters in a dictionary word with similar-looking numbers or special characters, like replacing ‘a’ by ‘@’, ‘i’ by ‘!’, ‘S’ by ‘$’, ‘S’ by ‘5’, etc. [Find more of such possible replacements, and make a list of the replacements you have considered in your program. **Submit** this list in a doc file].
- Password based on combining any of the above techniques, like ‘T@ctac1%’ [cat -> catcat -> tactac -> Tactac -> Tactac2 -> Tactac2% -> T@ctac2%]

Write a computer program with the following input and outputs:
Input: The dictionary word

Output: A list of common passwords, which could be created by using different variations of that dictionary word. Such variations should include at least the above-noted techniques commonly used by people in creating passwords.

- Use this program to enhance your list of common dictionary words, by adding their variations to that list. You can call it Enhanced List of Common Passwords 1.0 (ELCP 1.0).

Task 2 [Marks: 40]

Assume that a user needs to provide the following personal information during creating his/her online account with your organization (before password creation for that account):

i-a) First Name
i-b) Last Name
ii) Date of Birth (*Format*: MM/DD/YYYY)
iii) Telephone Number (*Format*: xxx-xxx-xxxx)
iv) Mailing Address
Format:
Street Information (Street number and name): _____, APT No. (if applicable): _____
City: _____ , State: _____, Zip Code (first 5-digit): _____
v) Email ID (it will be also used as a "User ID" for the account).

Users often create password based on their personal information, which include but are not limited to the following:

- First Name
- Last Name
- Date of Birth (DOB)
- A part of DOB, like the year of birth (1992 or just 92)
- Telephone Number
- A part of Telephone Number, like the first 6 digits or last 4 digits
- Street Number from Mailing Address
- Street Name from Mailing Address
- Apartment Number from Mailing Address
- Name of City from Mailing Address
- Name of State from Mailing Address
- Zip Code from Mailing Address
- Part of Email ID created by user, like for person@gmail.com: "person" could be a part of password; for person_gemini@gmail.com: "person" *and/or* "gemini" could be a part of the password.

Sometimes, users use a different variations of the above personal information to create their password, which include but are not limited to the following: Password based on replacing letters in a word with similar-looking numbers or special characters, like replacing 'a' by '@', 'i' by '!', 'S' by '$', 'S' by '5', etc. [Find more of such possible replacements, and make a list of the

replacements you have considered in your program. **Submit** this list in a doc file. To be clear, you will need to submit just one such list for Task 1 and Task 2].

Write a computer program with the following input and outputs:
Input: User's Personal Information (needed to provide for creating an online account with your organization: see above)
Output: A list of common passwords for that user, which could be created by using different variations of that information. Such variations should include at least the above-noted techniques commonly used by people in creating passwords.

- Use this program to enhance your list of possible passwords based on a user's personal information, by adding their variations to that list. You can call it Enhanced List of Common Passwords 2.0 (ELCP 2.0).

Task 3 [Marks: 30]
Create a user interface for online account creation (i.e., sign up) with your organization, that asks users to provide personal information noted in Task 2. After providing this information, the user is moved to the next page for password creation.

Once password is created, your program checks if it matches with any common password included in your ECLP 1.0 or ECLP 2.0. If a match is found, the interface alerts users about the vulnerability of their password to dictionary attack or targeted guessing attack with easy-to-understand explanation of why their password is vulnerable to such an attack [Example: "Your password is vulnerable to dictionary attack since you used the dictionary word: *'cat'*, or a variation of this word in your password"; OR "Your password is vulnerable to targeted guessing attack since you used your *mailing address* or a part of it in your password"]. Then, the interface prompts the user to create a new, stronger password.

The user-created password that is included neither in ECLP 1.0 nor in ECLP 2.0 would be accepted by the system, which would lead to the completion of account creation process at user's end.

B. Secure Storage [Marks: 30]

After the user creates a password that is not vulnerable to dictionary or targeted guessing attack, perform the following operations:
- Generate a 'salt'.
- Concatenate the 'salt' to user's password.
- Compute the hash digest of 'password.salt' [here, '.' represents a Concatenate operation]. You can use your hashing algorithm from Assignment 3, or any other standard Cryptography hash function of your choice.
- Store the hash digest in a password storage file.
- Store the 'salt' in a separate file.

C. Secure Login [Marks: 35]

Create a login interface where a user would enter his/her username (email ID) and password.

Identify if a user's password entered during login matches with his/her password created during sign up. In other words, a successful login would require a match between Hash (login-password.salt) and Hash (sign-up password.salt).

If a user enters a wrong password during login, he/she could try again. After a certain number of consecutive failed login attempts (3 consecutive failed login attempts for this assignment), the user would be locked out for a certain period of time (2^i [i = 0, 1, 2,..] minutes for this assignment) before he/she could attempt to log in again. See the login sequence below for a clearer understanding of lockout rules for this assignment:

1. Failed attempt
2. Failed attempt
3. Failed attempt
   Locked out for 2^0 = 1 minute
4. Failed attempt
5. Failed attempt
6. Failed attempt
   Locked out for 2^1 = 2 minutes
7. Failed attempt
8. Failed attempt
9. Failed attempt
   Locked out for 2^2 = 4 minutes
10. Failed attempt
11. **Successful attempt**
12. Failed attempt
13. Failed attempt
14. Failed attempt
    Locked out for 2^0 = 1 minute

Submission and Demonstration

- You will need to submit the noted deliverables including a working version of your code through email to GTA of this course (Manazir Ahsan, email: manazir.ahsan@aggiemail.usu.edu) before **11:59 PM on Saturday, November 09**. If needed, add additional instructions for running the code in a 'Read Me' file.

- One submission is required from each group (all group members need to be cc'd in the submission email). See Late Submission Policy in course syllabus.

- The subject-line of the email: **CS 5460: Assignment 4 Submission: <Group Name>**

- Each group, with all members present, will need to demonstrate the code on **Monday, November 11**. Attendance during demonstration is required to gain marks.