CS 370 Introduction to Security **Week 4: Problem Set 4**
Instructor Name: Rakesh Bobba

Introduction

The purpose of this assignment is to help you gain a better understanding and insight into user authentication concepts and the pros and cons of password-based authentication covered about in Week 4.

Before beginning make sure you have watched the lecture videos on the following and completed the associated practice quizzes.

- User Authentication
- Passwords: Pros and Cons I
- Passwords: Pros and Cons II
- Passwords: Pros and Cons III

Also make sure you have read this week's assigned reading from the textbook.

Questions

Please answer the questions below.

User Authentication and Passwords

Q1[10 pts]: You are designing a password system with randomly selected passwords. The alphabet for the passwords is the set of alphanumeric characters in English both upper and lower case and the integers 0-9. You are told that the attacker can make 250,000 guesses each minute.

a. If the passwords are 7 characters long, how long until the attacker has a 50% probability of correctly guessing user's passwords in an offline attack.
    a. P = (TG/N)
    b. $0.5 = (T * 250000 / (26 * 2 + 10)^7)$
    c. T = 13.4 years
b. How long do the passwords need to be to ensure that the 50% success rate is not reached until after 2 years?
    a. P = (TG/N)
    b. $0.5 = ((2 * 365 * 24 * 60) * 250000)/62^x$
    c. x = ceiling(6.539) = 7 characters for >= 2 years.
c. If the users select their own passwords, does this affect the relevance of your calculations from parts (a) and (b)? Explain your answer.
    a. The calculations above will not be accurate any longer, as the users will be allowed to enter passwords longer or shorter than 7 characters. If these are longer, then it will take a longer pure brute force to solve these. If they are shorter, it could decrease the time needed to solve them to (theoretically) mere seconds (for password 'a' for example).

Q2 [4 pts]: iPhone 6 includes a fingerprint scanner which the user can choose (not) to use. Do you think activating fingerprint scanning would increase the security of the cellphone? Why or why not?

- Given that apple does not use this as a second factor of auth, but instead for the iphone 6 family, they let the passcode take precedence over the finger print (center button press once on lock screen), the fingerprint sensor is merely a second way that a person can gain entry to a phone.
- These sensors are usually hacked with reasonably simple tricks given the security claims about them:
  - [TouchID, iPhone 5s] https://www.fastcompany.com/3018074/the-truth-about-the-newest-iphone-fingerprint-sensor-hack-and-why-you-sho
  - [FaceID, Many] https://www.forbes.com/sites/daveywinder/2019/08/10/apples-iphone-faceid-hacked-in-less-than-120-seconds/#474e12e721bc
  - [Samsung S7] https://boingboing.net/2019/06/12/artscrafts-bypass-a-fingerpr.html
- Based on the fact that there are likely working exploits for this, I would not add TouchID. Even if enabled, it is an alternative to the passcode, thus not adding much second factor auth.

Q3 [3pts]: Bloom filter is an efficient way to preemptively reject bad passwords with high efficiency, but it has a false positive rate (incorrectly rejecting good passwords). What can you do to decrease the chance of a false positive?

- Of many things that are doable to reduce these False Positives, one could implement the bloom filter with a hashing algorithm that has an output space closer to that of the input space (higher number of out bits, less pigeonhole principle), to reduce hash collisions. Fewer different hash functions working against the same array will also help, as there will be fewer bits that are 'turned on' as a result of {h_a(bad_pass), h_s(bad_pass) ...} that conflict with h_b(good_pass).

Q4 [3 pts]: Why will a bloom filter never give a false negative (accept a bad password)?

- This is because a 'bad password' will always be in the array (by definition of a properly set up Bloom Filter). Therefore, if a bad password is passed, it is guaranteed to hash to the same 'bits' in the array (property of a hash).

Q5 [3 pts]: It is common practice not to store user's password in clear text. However, if an attacker has seized control of the password database, he is likely already capable of modifying any user data on the site as an administrator. Why bother hashing the passwords then?

- Hopefully the db is encrypted in a way even an admin cannot view the passwords but can work with them while they are encrypted. This is closer to a proper implementation (though most aren't like this). This would mean that despite his ability to modify it, he will not be able to read it.

- It is worth hashing these things for the transit process to the db as well as keeping them safe at rest. If the site does not use SSL (or uses buggy SSL, for example) the passwords would be vulnerable.

Q6 [3 pts]: It is common practice to salt the user's password in addition to hashing. What attack does this practice prevent?

- This prevents a dictionary bruteforce attack as the hashes of the same value are different when salted and not salted, and the attacker would only have a dictionary on hand of values without that salt (it is hoped).

Q7 [4pts]: Does a "salt" used in password hashing need to be kept secret? Why or why not? Compare and contrast "salts" and "initialization vectors (IVs)" used in CBC encryption mode.

- It would not hurt. But these salts only prevent the attacker from arriving to the database with a dictionary or table of many, many hashes and cracking most of the passwords on your site within minutes. Once the attacker has the db, they will inevitably have any sat you use too – they will have access to the code that you are running on the server. Therefore, it is hard to keep the salt safe, and even if you do, it would not serve much purpose.
- If the salt is public, the attacker would still have to compile a dictionary of values, so it's best for it not to be public, either.
- The IV for CBC mode is also something that does not *need* to be kept private, like the salt. Both the CBC IV and Salt should be different each time the algorithm is run. They both serve as things that modify the 'plaintext' before it gets converted to 'ciphertext' - they differ, though in that the IV changes the entire plaintext, whereas the salt is typically just appended.

Submission Details

Submit a PDF file with the questions and your corresponding answers.
The assignment is worth 30 points. It is due Saturday of Week 4 at Midnight.