CSC4222/6222: Assignment 2

Due at 11:59 pm, Oct. 4

Part I: Questions

1. Assume that passwords are selected from four-character combinations of 26 alphabetic characters. Assume that an adversary is able to attempt passwords at a rate of one per second.
   a) Assuming no feedback to the adversary until each attempt has been completed, what is the expected time to discover the correct password?
   b) Assuming feedback to the adversary flagging an error as each incorrect character is entered, what is the expected time to discover the correct password?
2. It was stated that the inclusion of the salt in the UNIX password scheme increases the difficulty of guessing by a factor of 4096. But the salt is stored in plaintext in the same entry as the corresponding ciphertext password. Therefore, those two characters are known to the attacker and need not be guessed. Why is it asserted that the salt increases security?
3. a) In the context of access control, what is the difference between a subject and an object?
   b) What is the difference between an access control list and a capability ticket?
4. UNIX treats file directories in the same fashion as files; that is, both are defined by the same type of data structure, called an inode. As with files, directories include a nine-bit protection string. If care is not taken, this can create access control problems. For example, consider a file with protection mode 644 (octal) contained in a directory with protection mode 730. How might the file be compromised in this case?

5. Please describe the concept of the following methods of Threats & Attacks and give a countermeasure method.
   A. Denial-of-Service
   B. Correlation and Traceback
   C. ARP Spoofing
   D. IP Spoofing

6. Can you "decrypt" a hash of a message to get the original message? Explain your answer. What potential threats exist when you find out that, for every String A, you can find a String B such that H(A) = H(B)?

7. Please use a diagram to show how the process for the following scenario:
   Alice wants to send a file **M** to Bob. She wants to guarantee the **confidentiality** and **integrity** during the transmission. Besides, Alice also wants to demonstrate that **this message is sent from Alice**.

Please help Alice to design an Encryption System with diagrams.

8. Explain how signatures can be used for malware countermeasure. Give an example to support your explanation.

9. Assume passwords are limited to the use of the 95 printable ASCII characters and that all passwords are 10 characters in length. Assume a password cracker with an encryption rate of 6.4 million encryptions per second. How long will it take to test exhaustively all possible passwords on a Unix system?

10. In the context of biometric user authentication, explain the terms: enrollment, verification, and identification.

11. In the traditional UNIX file access model, UNIX systems provide a default setting for newly created files and directories, which the owner may later change. The default is typically full access for the owner combined with one of the following: no access for group and other, read/execute access for group and none for other, or read/execute access for both group and other. Explain the advantages and disadvantages of each of theses cases, including an example of a type of organization where each would be appropriate.

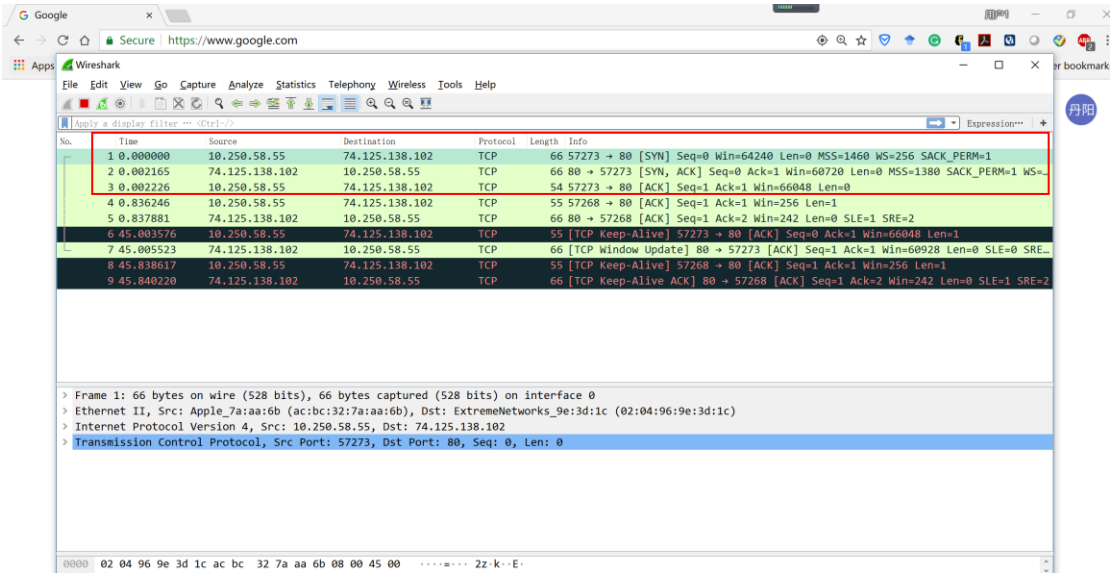
12. Name the five layers in TCP/IP Internet protocol stack. Briefly explain the main functionalities of each layer; identify the address mechanisms of each layer if any.

Part II: Wireshark & Programming

13. Use Wireshark to monitor the TCP to capture packets showing the TCP shaking hands process.

Step1: Download wireshark at https://www.wireshark.org/.

Step2: For example, use capture function to capture packets that specify the hand shaking process.



Requirements:

a. Student firstly needs to figure out the IP address of the csds.gsu.edu.
   Then, students are required to use wireshark to capture the TCP shake hands process packets as aforementioned.
b. Instead of the screenshot, student needs to upload xxxx.pcapng file (.pcapng file can be acquired from wireshark).
c. Specifically, students with same IP address will be regarded as copied work and get 0 as grade.

14. Password Salt System Implementation and Brutal Force Cracker

A. Implementation of the Password Salt System

In this section, students are required to implement a password salt verification system. A salt is stored in the database and added to the hashing process to force the uniqueness of the password, which is easy to verify and can increase the complexity without increasing user requirements. The salt does not need to be kept secret and the extra security which comes in with the salt is that it even maps same passwords to different hashes depending on the salt. This renders the dictionary attacks useless which are otherwise very effective in password cracking.

With the given UID.txt (UID stands for "Unique Identifier") and Hash.txt files, students need to implement the verification system, such that the given example of the password and salt can match with the hash value in the Hash.txt file. For example, in your UID.txt file, the first UID is 001, and in your Password.txt file, the password is 0599, and in your Salt.txt file, the salt associated with the first UID is 054. When applying the **MD5 Hash Function** with the **encode format as 'utf-8'** as shown in the figure, the expected output should be 4a1d6f102cd95fac33853e4d72fe1dc5 (See the first line of your hash.txt file). It is worth to mention that, the concatenation between password and salt needs to be in the format of (password||salt). For example, with the aforementioned input, the concatenation result will be 0599054. Note that, 0 should not be omitted.

Requirement for the designed system:

```
def computeMD5hash(my_string):
    m = hashlib.md5()
    m.update(my_string.encode('utf-8'))
    return m.hexdigest()'
```

1) The designed verification system should be able to correctly verify the example shown above. When the input is correct, the system will output a String "The input password and salt matches the hash value in the database". Otherwise, the output should be "The input password and salt does not match the hash value in the database".
2) Password_Salt_Helper.pdf gives a java template of the verification system. It is your choice to use the template and the programming languages (e.g., Java, Python).

B. Implementation of the Cracker System

To reduce the complexity for cracking the password and salt, the passwords are randomly set in the range of [0000, 1000], while the salt is randomly set in the range of [000,100] for each UID. One easy idea to implement a cracker system is to brute-forcely try all possible combinations of password and salt for one UID. As the Hash.txt and UID.txt files are given, students are requested to implement a cracker system which could find the correct password and salt for a specific UID.

Requirement for the designed system:

1) For a specific UID, the cracker system can output the correct password and salt value. For example, when input the UID as 001, the output should be "password: 0599; salt: 054".
2) Password_Salt_Helper.pdf gives a java template of the cracker system. It is your choice to use the template and the programming languages (e.g., Java, Python).

Submission Requirements:

1) The report should firstly describe how these two systems are designed; secondly, the report should include the set of passwords and salts for **ten** different UIDs. Source code and screenshot are required.

2) For undergraduate students, the verification and cracker systems can be designed separately. For graduate students, the cracker system should include the function of verification system.