Rafid
Shaon
CSC 4222
Oct 04 2022

CSC4222/6222: Assignment 2

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?

   a. The expected time is 26^4 / 2 or 228,488 seconds.

   Since there are a total of 26^4 passwords, the adversary can attempt half of the passwords on average in order to guess the correct one.

   b. The expected time is 52 seconds.
   Since there are a total of 26 characters, every character can be attempted 13 times (26 / 2). By flagging the characters as correct or incorrect, the time to find the correct four-character password is 52 seconds (13 * 4).

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 salt increases security?

The role of the salt is to add another layer of protection on passwords in order to increase the time it takes to be cracked. Having a unique salt will prevent the use of rainbow and hash tables, making the attacker spend more time computing the hash while factoring in the salt within the time to guess the password.

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?

   a. A subject is an entity capable of accessing objects with three classes (owner, group, world). An object is a resource to which access is controlled. The entity is used to contain and/or receive information.
   b. An access control list is a list of users with their permitted access rights. This gives users certain access rights to a particular object's resources. The transfer of rights is not allowed in this system. A capability ticket allows users to give out tickets to other users. This gives users in the system the authority to lend a

certain number of tickets to others based on where they are in columns and
rows.

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?

A file can be compromised based on their directory permissions. The owner's and
group's access rights are different when compromising these files. The owner has
the right to write certain files inside a specific directory where the group has no
rights to modify or delete the file, and the owner can give certain group members
rights to add and move these files.

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

   a. Denial of Service: the interruption or degradation of a data service or
      information access via sending large packet numbers to the provided service
      host
         i. implement a defense mechanism to guard against attackers in order to
            protect one's availability to their information
   b. Correlation & Traceback: the integration of multiple data sources
         i. provide a service to detect attacks; this allows one to filter and trace the
            source of the attack and have a way to block the attack
   c. ARP Spoofing: poisoning an arp cache by sending gratuitous arp replies
         i. use a virtual private network (VPN)
   d. IP Spoofing: an attacker's attempt to send packets from one IP address
      appearing to originate at another
         i. have an encryption protocol for IPS to prevent the traffic between
            enterprise servers to mask the IP address

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)?

No, you cannot "decrypt" a hash of a message to get the original message. A hash of a
message is a one way transmission function that allows the sender and the receiver
to "decrypt" the original message through a shared secret key. If the hash has been
changed, then the original message will also change, creating an overall new

message.

A potential threat when finding that for every String A, there is a String B through H(A) = H(B) is that an attacker can steal this information and use it to decrypt messages that were sent via hash. This allows them to read the sent messages without needing to know the original message.

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.

Thought Process:
- Alice uses her private key to encrypt and hash file M
- Alice sends it over to her shared key with Bob (public key)
- Bob uses shared key to receive file M
- Bob uses his private key to decrypt file M and verify that file has not been altered & Alice was the one that send it

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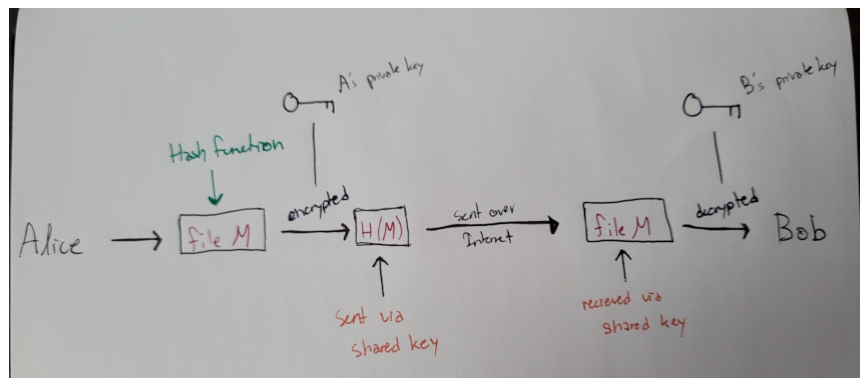
8. Explain how signatures can be used for malware countermeasure. Give an example to support your explanation.

Signatures can be used for malware countermeasures by analyzing an object and comparing it to a database of signatures. These signature databases can provide information on unique, common identifiers to new viruses. For instance, a string with a sequence of instructions specific for each virus makes a signature a fingerprint for the virus. Although this is not a digital signature, the database keeps records of the virus based on its signature.

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?

It would take about 296,653 years.

Assuming that there are 95^{10} possible passwords and the encryption rate is
$6.4 * 10^6$ per second

possible passwords / encryption rate

95^{10} / $6.4 * 10^6$ = 9,355,264,675,599.67041015625 seconds

(((seconds / minutes) / hours) / days)

(((9,355,264,675,599.67041015625 / 60) / 60) / 24) / 356)

= 296,653.4968163264336046502409944 years

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

   1) <u>Enrollment</u>: creates an association between a user and their biometric characteristics
   2) <u>Verification</u>: when authenticating, this confirms if the claimed user is the actual user or not based on the user's biometric characteristics
   3) <u>Identification</u>: when authentication, this confirms known and unknown users based on the user's biometric characteristics

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.

   1) <u>no access for group and other</u>
       i)   The owner is able to access newly created files and directories, only allowing groups and other groups to access these rights by granting them permission.
       ii) <u>Advantage</u>: files are protected from unauthorized access and modification from outside attackers and unknown users; only the owner can access the file
       iii) <u>Disadvantage</u>: to gain access to files, the group or other group must request for permission to access due to the files being restricted
       iv) <u>Example</u>: keeping the work used by the government systems private
   2) <u>read/execute access for group and none for other</u>
       i)   The owner and their groups are able to access newly created files and directories, not allowing other groups to have these access rights.
       ii) <u>Advantage</u>: file sharing is enabled between this particular group; only they are able to access these files
       iii) <u>Disadvantage</u>: multiple users from the same group may add, remove, or modify these files, increasing the risk of damaging said files
       iv) <u>Example</u>: having a software company connect their database with their

clients; the people assigned to a specific project only need access to the database to get the information that they need

3) read/execute access for both group and other

    i) All users are able to access newly created files and directories.

    ii) Advantage: all users have access to these files; this improve work efficiency since the information and data are available at any time

    iii) Disadvantage: users with no access to these files are able to get permissions for said files; data can get leaked to their company's competitors

    iv) Example: sharing a link to a Google document where anyone that has access to the link can look at and modify it

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.

1) Application: the supporting network applications
   - uses different applications and protocols to transfer information from one end of the system to the other end
   - some protocols include HTTP (transfers web documents), SMTP (exchanges electronic mails), and FTP (transfers files from one node to another)

2) Transport: the process-process data transfer
   - transports the messages of the application layer between two endpoints
   - some protocols include TCP (guarantees the delivery of messages) and UDP

3) Network: the routing of datagrams from source to destination
   - moves packets of data (or datagrams) between any two hosts in the network
   - main protocol is the IP, which defines the datagram

4) Link: the data transfer between neighboring network elements
   - moves packets from one node to the next node through an optimal route; also known as the datalink layer
   - main protocol is the PPP, which finds the optimal route

5) Physical: the bits "on the wire"
   - moves the individual bits within the frame from one node to the next node via physical transmission media
   - different physical media uses different protocols

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.
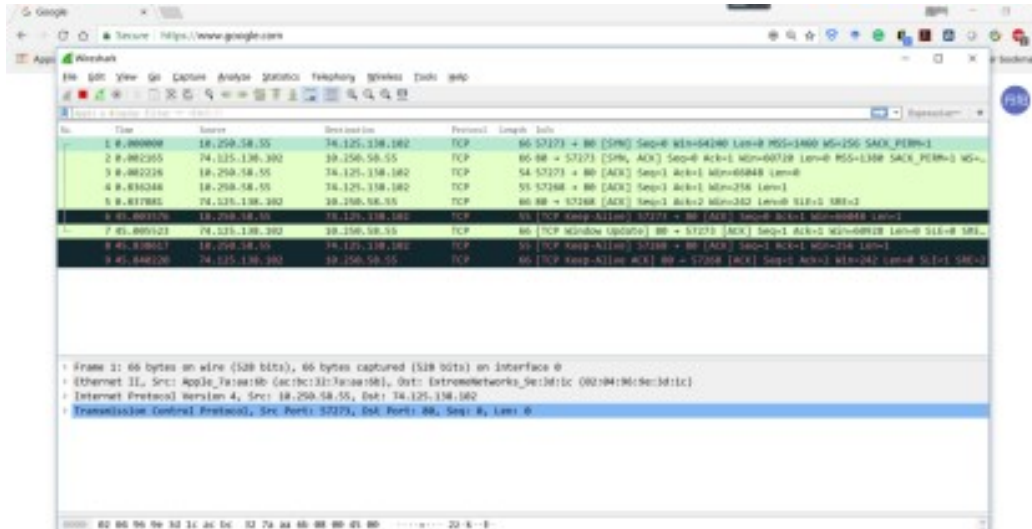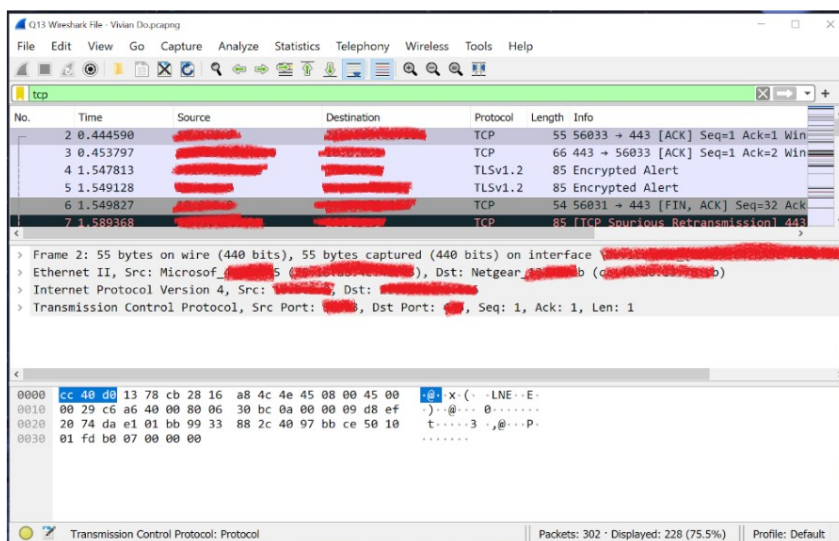

Requirements:

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.

Screenshot will be printed below. My *.pcapng* file will also be submitted in the Dropbox.

Screenshot:

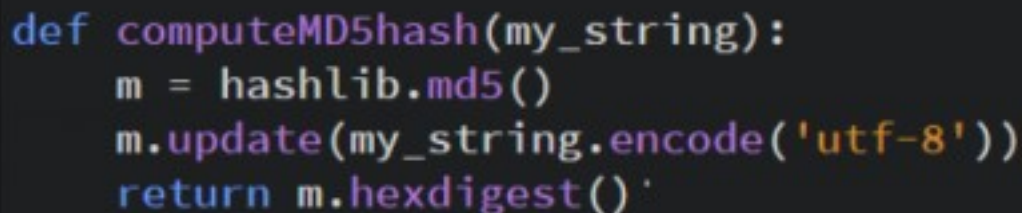14. Password Salt System Implementation and Brutal Force Cracker

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 <u>001</u>, and in your Password.txt file, the password is <u>0599</u>, and in your Salt.txt file, the salt associated with the first UID is <u>054</u>. When applying the **MD5 Hash Function** with the **encode format as 'utf 8'** as shown in the figure, the expected output should be <u>4a1d6f102cd95fac33853e4d72fe1dc5</u> (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.

Although not working 100% of the time, I've used three programs to complete this problem: Q14A.java, Q14B.java, and Data.java. These source codes will be submitted in the Dropbox as well. For part A and Part B, there are two identical methods used to create a hash function MD5 within the charset UTF-8 and to compare a system-generated hash with the hashes from the text files (Data) respectively.

a. Q14A.java is my Password Salt System. I made a way for the user to input their User ID, password, and salt. The system then uses the sum of their inputted password and salt to generate a new hash; this is then converted into the utf-8 character set of bytes through the MD5 hash function. This new hash is then compared with the hash in the system/database (Data.java) through a function; if they are have the same properties(password/salt) or not, then it verifies the user that the password and salt matches the UID, and vice versa if it does not match through a message.

b. Q14B.java is my Cracker System. Using the same Scanner, the user is able to input their desired UID. The brute force method (findPW()) will then be called. Two nested for-loops are created to loop through the range of password and salt combinations and are then initialized to a temporary string. Another method is called to compare the hash in the system/database (Data.java) and the temporary string. If the UID matches the temporary string, the password and salt will print; an error message will print if the UID does not match.

- Data.java consists of multiple different methods. The *readFiles()* method opens each text file and pushes every content inside into an array (one per text file). The *set* methods return the arrays with an element *i*, and the *get* methods pushes the contents of the file into an array of elements. The last three methods *uidNum()*, *hashNum()*, and *passNum()* return the size of each array respectively.