Faiza Zubair

1).Devising an application (e.g. devising a game or server client application) with some valuable measures inside (passwords, precious information). The more creative and amazing the application is, the better. (As an example, imagine that our application is a game which allows the user to input a sequence of five numbers and tells, as an output, if the sequence matches another sequence, named the jackpot sequence, which is hard-coded in an array variable inside the application source code.)

The objective of this project is to evaluate the effectiveness of a specific source code obfuscation technique, namely string encryption, in protecting sensitive information from malicious attacks in Android applications. The project will involve the following steps:

Application Development:

I am using Android Studio and Java to develop an Android application that incorporates valuable measures, such as a login system with encrypted passwords.

I have created a user-friendly login system where users can register, log in, and access protected resources.

Stored user passwords securely using encryption techniques within the source code.

2. Carry out an attack task by writing a piece of code. Reverse engineer or decompile the program to access the part which you expect to be hidden (For the aforementioned application the attack task is to determine the jackpot sequence). Report how much time you need to understand the expected hidden.

I have used online java compiler named jadx compiler to decompile the code by giving my application's bytecode file (apk. File)

Analyzing the Decompiled Code: While analyzing decompiled code what I have found it gave me clear structure of code.

Identify Encryption Mechanisms: While analyzing password encryption mechanism ot is clearly understandable The hashFunction() method correctly uses the SHA-256 algorithm to generate the password hash.

Understand Password Handling: In decompiled code it is clearly shown that simple hashing mechanism is used to encrypt and store passwords. Here's a breakdown of the password encryption mechanism used:

Hashing Algorithm: The SHA-256 hashing algorithm is used, which is a widely used cryptographic hash function known for its security and collision resistance.

Salt: To enhance the security of the hashed passwords, a randomly generated salt value is used for each user. The salt is a unique value that is combined with the password before hashing.

Hashing Process: When a user registers, the plain-text password is combined with the salt value. The combined string is then hashed using the SHA-256 algorithm to generate a secure hash.

Storage: The hashed password, along with the salt value, is stored in the database. The salt is typically stored alongside the hashed password.

Authentication: When a user attempts to log in, the entered password is combined with the stored salt value. The combined string is hashed using the SHA-256 algorithm, and the resulting hash is compared to the stored hashed password. If the hashes match, the password is considered valid.

By using a salted hash, we add an extra layer of security to the password storage. The salt value ensures that even if two users have the same password, their hashed passwords in the database will differ.

3). Obfuscate the application with an approach you implement yourself. (As an example, we can use array transformation methods learned in the class to hide the content or usage of the array). Write the approach in another program. The program should read a program as an input, find the part you intend to obfuscate and present an obfuscated version of the code as an output.

I tried to apply **string encryption obfuscation technique** to hide the encryption method used in the application but did not do it. But you can see my idea

Here's an example of applying string encryption obfuscation to the application.

Identify the Encryption Part:

In the User model, the encryption part is located in the setPassword method where the password is combined with the salt and hashed.

Implement String Encryption:

Here's a basic implementation of the encryptString and decryptString functions using use XOR-based string encryption technique (Data obfuscation techniques which aims to secure just sensitive part):

```
public class EncryptionUtils {
    private static final String ENCRYPTION_KEY = "secretkey";

    public static String encryptString(String input) {
        char[] key = ENCRYPTION_KEY.toCharArray();
        char[] data = input.toCharArray();
        StringBuilder encrypted = new StringBuilder();

        for (int i = 0; i < data.length; i++) {
            encrypted.append((char) (data[i] ^ key[i % key.length]));
        }

        return encrypted.toString();
    }

    public static String decryptString(String encrypted) {
```

```
        return encryptString(encrypted); // XOR encryption is symmetric
    }
}
```

Encrypt Password Encryption Code:

In the User model, I modified setPassword method to use string encryption on the password:

```
public void setPassword(String password) {
    // Encrypt the password
    String encryptedPassword = EncryptionUtils.encryptString(password);

    this.passwordHash = encryptedPassword;
}
```

In login Activity I applied Code obfuscation technique named as identifier renaming. I changed the name of loginActvity class to A. and all variables to simple string names .

Other technique Which I am using to make code more complex is control flow obfuscation by adding unnecessary branching and altering the control flow, it makes the code more complex and difficult to understand, thus potentially confusing or hindering reverse engineering attempts.

The purpose of such control flow modifications is to make the code more convoluted and less predictable, thereby increasing the difficulty for an attacker to understand the code's logic and potentially disrupt any reverse engineering attempts.

4).Repeat the attack task built in the second phase once more on the obfuscated program. How much time you spent on breaking the code and identify the hidden? Is it possible to break the application?

Again I used jadx to decompile code but obfuscated code in kind of difficult to understand for common user. It takes more time to decompile this code rather than implementing a new class. Because of obfuscation applied if takes time to understand.

5). Report Measurement of efficiency which means the delay to break the application in these two versions a. First, clear version of your application b. Second, obfuscated version of the application

In first version it took very less for me to decompile and understanding the code structure because I have some expertise on this but if someone don't have knowledge it can take a lot of time.

As a developer it is time taking to decompile obfuscate code because it take a lot of time to de-obfuscate. Rather than decompiling this code we can make new code instead in same time. But it can be beneficial if we don't know how to implement the functionality of the original code and try to obfuscate other's code to complete give task:D