DOP: / /2023 **DOS: / /2023**

Experiment No: 2.

Title: Android Security Case Studies.

Theory:

What Is Android Studio?

Android Studio is an integrated development environment (IDE) created by Google to help developers create Android applications. It provides a wide range of features that make it easier to write, test, and deploy applications to Android devices. Android Studio supports all the major mobile SDKs such as the Android SDK, the Android NDK, and the Google Play Services SDK. It also includes several writing and testing tools for things such as unit testing and version control. It can also be used to build and publish Android applications to the Google Play Store. Additionally, Android Studio includes a graphical user interface for managing the emulator, creating activity layouts, and creating customized UI components.

The Android Concepts:

We first describe Android's main concepts, before presenting the challenges related to analyzing the platform with respect to security. Note that there does not exist a comprehensive document on Android's security concepts. The information is scattered throughout the Android developer's website. Android Components: An Android application consists of different parts, called components, having, according to its task, one of four basic component types. Activities are the presentation layer of an application, allowing a user to interact with the application. Services represent background processes without a user interface. Content providers are data stores that allow developers to share databases across application boundaries. Finally, broadcast receivers are components that receive and react to broadcast messages, for example, the Android OS itself sends such a broadcast message if the battery is low. Each component of an application runs as a separate task, making an Android device to a large distributed system, even if all processes are running on the same device.

How Android To Users:

Android provides security to users through a variety of mechanisms, including: Application Sandboxing: Each Android application runs in its own sandbox, isolated from other applications and the system, which helps to prevent unauthorized access to data and system resources. Permissions: Android uses a permission-based model to allow or deny access to specific features and data on a device. Users are prompted to grant or deny permissions when installing an application. Security Updates: Android releases security updates on a regular basis to patch vulnerabilities and protect against known security threats. Google Play Protect: It scans apps and checks the device for potentially harmful apps in real-time. Encryption: Android devices can be encrypted to protect data stored on the device. Built-in malware protection: Google Play store has built-in malware protection to protect users from downloading malicious apps.

Main 4 Security Techniques Used By Developer:

1. Protect Your App's Transport Layer:

One of the first things an attacker will look for when targeting an Android app is to see if they can intercept any of the data passing between it and your server's backend By eavesdropping on those communications, they can tell an awful lot about your app. And if you're really unlucky, they might even be able to use the data to figure out how to impersonate your app and gain inappropriate access to server-side data.

2. Make Authentication Bulletproof:

Besides your app's data streams, the next most common attack vector to eliminate is any weakness in its authentication methods. The trouble with doing that is your users. By that, I mean that you need to make your app's authentication process as secure as you can without making it so onerous that your users will revolt (and if I had a dollar for every time a client asked me if 2FA was really necessary, I'd be retired by now). Regardless of how many times you have to answer the question, though, 2FA is both necessary and worth implementing. On top of that, you also need to pay attention to how you handle things like key exchanges. At a minimum, you should be using AES encryption to keep those transactions secure.

3. Guard Against Code Injection :

The next thing you should worry about is the public-facing parts of your app. Because most apps are interactive, they will provide users the ability to input data in one form or another. This can be through text-input fields like forms, or through direct data uploads for exchanging things like documents and pictures. And every time you add a user input feature, you should take great pains to make sure that nobody can use them against you. The first way to address this is to use proper input validation. If your app expects specific text in a field, make sure it won't accept anything else. You can do this by adding a pre-built text validation module or by building your own (if you have the time and inclination).If you plan to let a user upload pictures or other specific files, you must include an ability for the app to scan the uploaded file to make sure it is what it claims to be.

4. Minimize and Secure Client-Side Storage :

Last but not least, you should strive to build your app with the smallest local data footprint that's feasible to get the job done. That's because any data you store on a client device is outside of your control and is therefore vulnerable to external threats. If a user's device gets compromised by any attack that yields access to the data stored on it, you may inadvertently give the attacker a roadmap to steal information from your app.

Android Study:

We conducted our security assessment along with a security expert, the third author, to understand the implementation of some aspects of the Android security mechanism with the help of the Bauhaus tool-suite.

Issues: Starting from the documentation, we planned to focus on several aspects related to the implementation of permission checking. The process of permission enforcement consists of different steps, which we aimed to understand. After reading the security documentation, available at the Android website, we picked the Bluetooth API for further investigation. Other functionality of the platform could have been analysed in a similar way. The Bluetooth API documentation states that an application needs at least the BLUETOOTH permission to use the Bluetooth device. If a program wants to administer the Bluetooth device, it needs the BLUETOOTH_ADMIN permission in addition. It is explicitly noted that one needs the former permission, to use the latter. During the security assessment we wanted to answer several questions the security expert had. We will discuss these in the following: Question 1 Where are permissions enforced within the Bluetooth API (enforcement points)? Question 2 Which permissions are enforced within the Bluetooth API (access control policy)? Question 3 Can we check whether an application needs the BLUETOOTH permission, in order to use the BLUETOOTH_ADMIN permission?

Conclusion: -

Thus we have studied about android, there various security and the history of the android security.