**Chapter 1 Introduction**

Software Reverse Engineering (SRE) is the practice of analyzing a software system, either in whole or in part, to extract design and implementation information.

**1.1 The Aim of The Project**

The aim of the project is to design and implement a development framework to be used by Android applications developers to make their applications more immune to reverse engineering and license tampering.

**1.2 Why Android and not iOS**

Protecting a software from reverse engineering is hard in general. Protecting Android applications from not being reversed is harder than iOS due to many reasons, the fundamental reason is that Android is an open source, anyone can look what’s inside it and try to analyze and understand it, unlike iOS, which is not. This makes Android more vulnerable to malware attacks. Other reason is that the Java bytecode is especially easy to reverse engineer. For example, the source code can be recovered by converting the APK to JAR file, then a java decompiler can be used to generate the code for the application. Moreover, in Android, other than google play store, it is possible to install the applications from unknown sources. But, in iOS, the apps can be only installed from AppStore. It is one of the major security breaches in Android. Due to various security breaches in Android, attackers already regard smartphone as the target to steal personal information using various malware.

**1.3** **Android Security Issues and attacks**

In order to strengthen Android against reverse engineering, we must understand it’s security issues first and understand the attacks it is experiencing.

**1.3.1 General Security Issues**

* In the latter part of 2010 and early 2011, a vulnerability issue was discovered in Android versions 2.2 and 2.3, respectively. The vulnerability is that an attacker can copy any file that is stored on the device’s SD Card without granting a permission or even without a visible cue that this is happening.
* The idea for android being open source itself is a problem; attackers can analyze each line of code to determine its weaknesses.
* Google play store is a bit of concern because of the relative ease of getting apps approved for sale. Malware apps can squeak through.
* In Android, it is possible to install the applications from unknown sources, like third-party android stores. It is one of the major security breaches in Android.

**1.3.2 Permission Escalation Attack**

It allows a malicious application to collaborate with other applications to access critical resources without requesting for corresponding permissions explicitly.

**1.3.3 Collision Attack**

Collision attack is a technique wherein two or more application share the same user ID so that they can access the permissions, which are granted to each other. For example. If application A has permissions to READ\_CONTACTS, READ\_PHONE\_STATUS and B has permissions to READ\_MESSAGES, LOCATION\_ACCESS, if both the applications use the same user id SHAREDUSERID, then it is possible for application A to use the permissions granted to itself and the permissions granted to B. Similarly, it is possible for application B to use the permissions granted to itself and the permissions granted to A. Every Android application has unique ID that is its package name. Android supports shared User ID. It is an attribute in AndroidManifest.xml file. If this attribute assigned with the same value in two or more applications, then they can access permissions granted to each other.

**1.3.4 Dangerous Permissions**

Dangerous Permissions can access critical resources of the mobile. Dangerous permissions can give the app access to the user's confidential data. If app lists a normal permission in its manifest, the system grants the permission automatically. If app list a dangerous permission, the user has to explicitly give approval for the app for the successful installation of the app. Example:

CONTACTS

READ\_CONTACTS, WRITE\_CONTACTS,

GET\_ACCOUNTS

LOCATION

ACCESS\_FINE\_LOCATION,

ACCESS\_COARSE\_LOCATION

SMS

SEND\_SMS, RECEIVE\_SMS, READ\_SMS,

RECEIVE\_WAP\_PUSH, RECEIVE\_MMS

STORAGE

READ\_EXTERNAL\_STORAGE,

WRITE\_EXTERNAL\_STORAGE

**References :**

reverse engineering

http://reversingproject.info/

\_\_\_\_\_\_\_\_\_\_\_\_\_

Android vs iOS

https://www.quora.com/How-resistant-are-Android-apps-to-reverse-engineering