A FRAMEWORK TO DETECT THIRD-PARTY CLONE APPS

[1]Manish Vishwakarma, [2]Bipin Tiwari, [3]Ashish Yadav, [4]Ravi Nagar
[1, 2, 3] B.E Student, [4]Assistant Professor
[1, 2, 3, 4] Dept of Computer Engineering
[1,2,3,4] Universal College of Engineering, Mumbai, India

Abstract: As we already know google play store is very known platform to download the various Applications. But there are various third-party marketplaces who provide the apps to the users. Major issues with the third-party marketplace are that they develop the apps which are almost similar to the apps which are already available on the google play store and then they upload the same app on play store because Google doesn't check similarity of apps rigidly. Therefore the number of duplicate appication on Google play store increases day by day which might cause some serious problems to the smartphone users. So to eliminate of this problem we are making a framework which checks the app developed by the developer before uploading on the Google play store. To do these we are using decompiler Apktool to match the similarity between the genuine app and the newly developed app. We will compare the codes of the genuine app and newly created app to a particular limit. If the newly created app crosses our set limit then we declare that app as a clone app. The motivation behind doing this is to make Google play store safe and convenient for the users to use.

Index Terms **–Smartphones, Repackaging, Encrypted Key, Privacy and Security.**

I. INTRODUCTION

In recent years mobile handheld devices have become the most important thing in our daily life. The mobile handheld device comes with the new and intelligent feature that attracts people towards them. Using smartphones people do their daily work very easily but on the other hand dependency on mobile phones becomes a major issue when it comes to security. Most of the smartphone users install the apps without knowing much about that app and they enter their personal information in that app. As we already know Android grows into the most popular mobile OS by global market share, Android-targeted attacks continue rising in both number and complexity. More than 85% of Android malware rely on application repackaging to spread too many genuine users. In this paper, we are making a framework to identify repackaged applications. Repackaged applications are nothing but the application whose functionality, code structure and GUI are almost similar to an already existing genuine application on Google play store. Many developers do this to save their time, efforts, earn more profit in less time and also to steal user's sensitive information. Due to continuously uploading repackaged applications on Google play store increases the load on the play store, complexity and it also becomes a difficult task for the user's to search the genuine application among the repackaged applications. Hence to deal with these problems, in our framework we maintain a database of genuine applications and will compare the code, structure, and logic of newly created applications with a genuine application. If the similarity is found above the considerable limit then we will declare that application as a repackaged application.

II. RELATED WORK

1. A Framework for Third Party Android Marketplaces to Identify Repackaged Apps. This paper was published in the year 2016 and author of these paper is Nia-Wei Lo, Shao-Kang Lu, Yohan Chuang. This paper was published on IEEE. Proposed work of these paper is a real-time App integrity verification system to check an uploading App is a genuine one repackaged one. Research gap of these paper is App is different from v1.0.0, for example, v1.2.3. In this kind of situation, our App integrity mechanism could not verify correctly.

2. Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces. This paper was published in the year 2012 and author of these paper is Wu Zhou, Yanjing Zhou, Xu Xian Jiang, Peng Ning. This paper was published on ACM Conference. Proposed work of these paper is implementing an app similarity measurement system called Droid MOSS. Research gap of these paper it has been reported that even in the official Android Market, there may exist malicious apps repackaged from other legitimate apps.

3. A Framework for Evaluating Mobile App Repackaging Detection Algorithms. This paper was published in the year 2013 and author of these paper is Hewing Huang Sensum ZhuPeng LiuDinghao Wu. This paper was published on Springer. Proposed work of these paper is to evaluate the obfuscation resilience of repackaging detection algorithms. Research gap of these paper is they have lack of a comprehensive analysis on algorithm accuracy.

III. PROBLEM STATEMENT

Needless to say Google Play store is the official market of Android apps and there are approximately 2.5 million apps available on the google play store. App developers earn money by selling apps through in-app billing and through advertisement. The apps which are more popular are disassembled by adversaries who then replace the advertisement and other in-app feature from the app to steal the profit of actual developer. Due to this the actual developer and smartphone user gets affected in terms of profit, sensitive information, their private schedule etc.

IV. REQUIREMENT

3.1 Hardware Requirement

- Laptops

3.2 Software Requirement

- APK Tools
- Decompilers
- IntelliJ

V. IMPLEMENTATION

We proposed a market framework which is based on the App verification mechanism we designed. In this framework, we have implemented an App verification process so that when someone wants to upload an App which is not verified by the official organization such as Google, the App should pass the verification process first. Only when uploading Apps are verified as equivalent to the original ones (from Google Play Store) or harmless ones, they are released onto the market. Android applications are installed in the form of application package files which is commonly known as APK files. APK files are basically container files that contain both Application codes, resources as well as the application manifest files. Normally APK files are in the form of Zip files and we can check it by extracting with any compression utility that supports the ZIP format.

In general, Apk files consist of following parts:

ANDROIDMANIFEST.XML	The manifest file in binary xml format.
CLASSES.DEX	Application code compiled in the dex format.
RESOURCES.ARSC	File containing precompiled application resources in binary xml.
RES/	Folder containing resources not compiled intoresources.arsc
ASSETS/	Optional folder containing applications assets, which can be retrieved by asset manager.
LIB/	Optional folder containing compiled code - i.e. Native code libraries.
META-INF/	Folder containing the manifest.mf file, which stores metadata about the contents of the JAR. Which sometimes will be stored in a folder named original. The signature of the apk is also stored in this folder.

Every APK file has an AndroidManifest.xml file represents the application's package name, version components and other metadata.

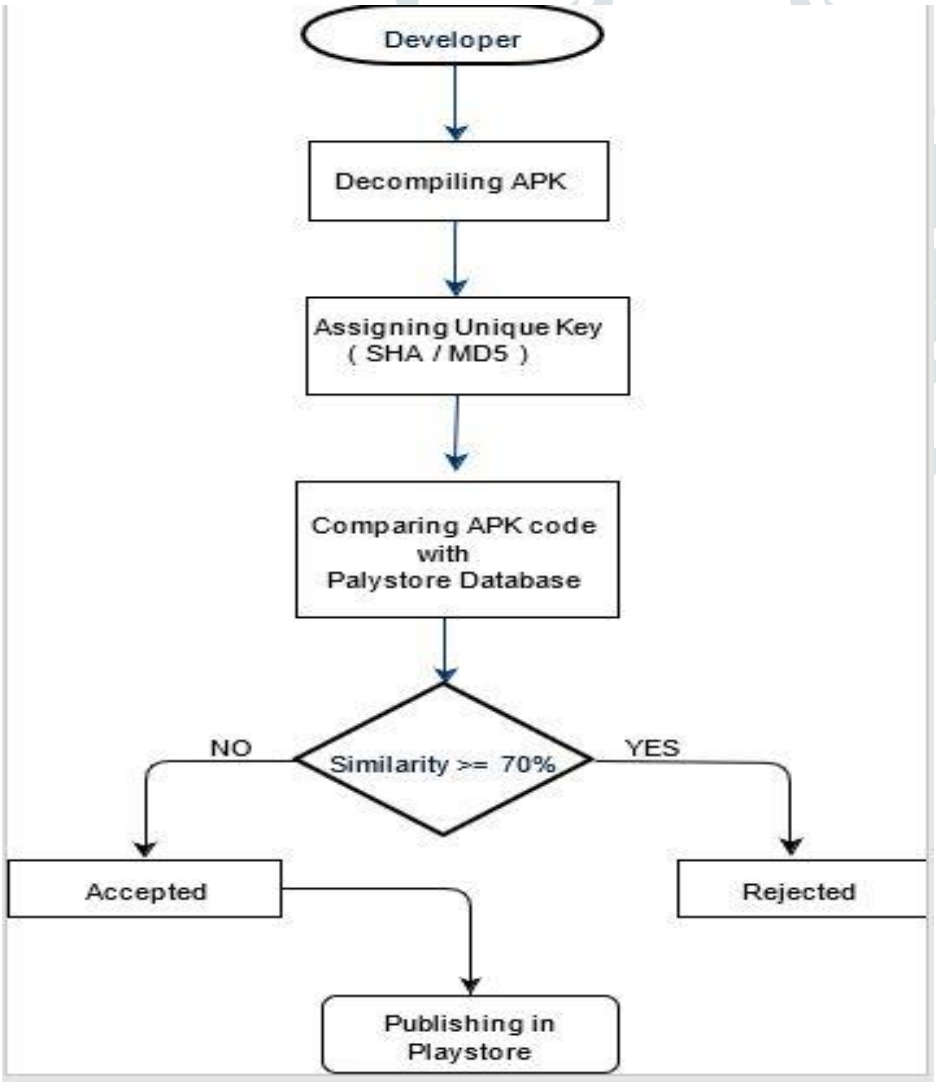
The working of the flowchart is as follows:

- **DEVELOPER:** Developer is the person who creates the application and uploads it on the Google Play store. So this is a general procedure which is followed by the developer but our project is like an intermediate between the developer's application and the google play store. Developer's application will be uploaded in our framework and from here it will undergo through series of operations.

- **DECOMPILE APK:** In this phase, we are decompiling the application of developer using a tool known as Apktool which is linked with IntelliJ software. Apktool is reverse engineering tool. This tool will decompile the application and shows all the components of the application which are mention above in the table.

- **EXAMINE APK INTEGRITY: -** The main task of our project starts from here. In this phase, we maintain a database of a genuine application and assign a key to every application and then we will compare the Apk of newly created application and Apk of a genuine application with the help of Apktool and IntelliJ software. It will compare both the application by checking code structure, logic, and GUI and shows the result accordingly. If the application is found unique then we will assign a unique key to the newly created application.

- **SIMILARITY**: - This phase is the last phase of our project. We will check the similarity of the applications on the basis of their behavior, code structure, and logic. After doing all these operations if we found newly created application is approximately 70% or more similar to a genuine application existing in our database then that application is declared as repackaged or copy application and will tell to do modification in that application. If an application is found unique then we will permit to upload that application on Google play store.



VI. CONCLUSION

At present Android applications become one of the vital things in our daily life. This is because the application is considered as a computer software package that performs a specific function directly for an end user or, in some cases, for another application. Hence the Android application development becomes the profession to earn money but on the other hand, it also becomes harmful for the users when it comes to protection of sensitive information of the user. So by developing this project, we will prevent repackaged (copy) applications from uploading on Google play store and we are trying to put our foot one step ahead towards the security and reliability of Android applications. We are also trying to make Google play store efficient, secure and simple for the user.

VI. REFERENCES

- Alexios Mylonas, Bill Tsoumas, Stelios Dritsas, and Dimitris Gritzalis, “A secure smartphone applications roll-out scheme,”Trust, Privacy and Security in Digital Business, vol.6863., 2011,pp.49-61.
- William Enck, Damien Octeau, Patrick McDaniel, and Swarat Chaudhuri, “A study of android application security,” SEC'11: proceedings of the 20th USENIX conference on Security, pp. 21-21, 2011.
- Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steven Hanna, and David Wagner, “A survey of mobile malware in the wild,”SPSM’11, pp. 3-14, October 17, 2011, Chicago, Illinois, USA.