Comparative Analysis of Android Applications in Unofficial and Official Markets

HyunSoo Lee*, Eul Gyu Im*
*Dept. of Computer and Software
Hanyang University, Korea,
E-mail : {evasioner, imeg}@hanyang.ac.kr

1. Introduction

In recent years, the popularity of smartphone equipped with Android operating system is increasing. Therefore, a number of smartphone users are threatened by many attacks including the leakage of personal information. Many different kinds of attack exist currently. For example, attacks like the small payments and the leakage of personal information through a message are possible. To achieve these kinds of attacks, attackers insert malicious code in famous benign applications, and repackaging them. As a result of repackaging attacks, normal users install repackaged applications which are made by attackers without doubt and the attackers can obtain personal information from smartphone users.

Many security analysts have been researching in order to respond and depend against these attacks. In this paper, we propose a method to check if applications are the repackaged applications or the normal applications. The first method is to check the signature. The second is to compare the hash value of fingerprint of applications which are downloaded from unofficial markets or the official market. If the attacker repackages an application, attacker have to resign the hash value of the application. Since it requires the resigned signature, it assumed to be different. We performed these two method to confirm whether an application is repackaged or not.

2. Related Work

C.Y. Huang et al.[1] proposed a method that displays a result as comma and of numeric type for whether benign or malware by comparing a generated feature vector through extracted information as extension name from the decompiled APK file and features as defined permission in AndroidManifest.xml with the extracted information from samples. Y.Zhou et al.[2] proposed TOP 20 graph of the permission list of most used permissions in the downloaded a total of 2,520 applications from Google Play Store and malicious applications.

Blasing, Thomas, et al.[3] obtained information of AndroidManifest.xml and the byte code. And they also collected system call trace using LKM (Loadable Kernel Module). Then, they proposed a method that expresses a histogram through obtained data from the dynamic analysis and the static analysis.

3. The Repackaging Testing Method

When an Android application is developed, it requires a key in order to use Google APIs. In addition, it requires a fingerprint with signature. The hash values of fingerprints of developed applications in the same company are equal. In case of applications repackaged by an attacker, the hash values of fingerprints are different from the original fingerprints. This is the reason why attackers have to re-sign modified applications.

Even though the period of use is not specified in an application, users can use the application, after checking the signature when the application is installed. The period of use is usually set to 27 years, and the signature of an application contains information such as “company name”, “organization”, “location”, etc. However, most developers do not insert information into applications. Therefore, when we checked applications, most applications have a value as “Unknown”. In this paper, we collected applications from *Google Play Store* [4] and unofficial websites for experiments. We collected 1,222 samples from the official market and 2,048 samples from the unofficial markets.

Because the names of samples which were collected from the unofficial markets may have meaningless words, we changed the names of application to their package names. We developed a tool (called *Jarsigner.exe*) to check whether each application is sign or not. We could obtained the hash value of fingerprint of each sample with this tool, and we stored these hash values and corresponding package name into a database. Then, we compared the hash values of fingerprints among the samples which have the same package name.

4. The Experimental Results

We performed two experiments: the first experiment is to check the signatures of applications, and the second is to compare the hash values of fingerprints.

First, we checked the hash values of fingerprints of applications with *Jarsigner.exe.* However, it is difficult to check a large amount of application samples with this tool. Therefore, we developed a novel tool can check about 3,200 applications which we had already collected. The tool we developed is an enhanced version of *Jarsigner.exe*.

After we collected the information from every sample using this tool, we produced the statistical results to compare the samples from the official market and the samples from the unofficial market. The result of the experiment is shown in Figure.1

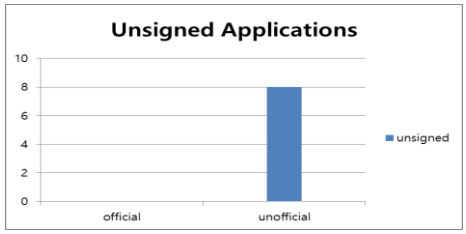

Figure 1. The Number of Unsinged Applications

Figure.1 shows that applications of official are all signed, while 8 applications of unofficial are not signed. This means that most applications in unofficial markets are re-signed after repacking.

In the second experiment, we compared the sign hashes of fingerprints of collected applications from both official and unofficial website. Based on the package name of applications, we compared the sign hashes. The result of experiment shows in Figure.2.

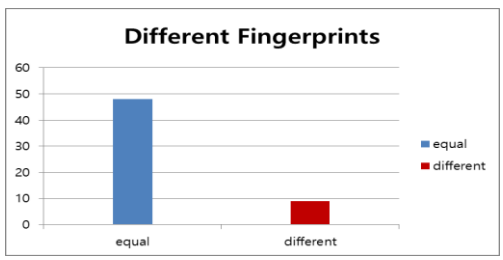

Figure 2. The Number of Different Fingerprints

Figure.2 shows that some fingerprints of applications have different hash values even though their package names are same. As a result, a total of 9 applications have different hash value in the same 57 applications. 9 applications are also suspected to be signed after repackaging. APK files require re-signing, otherwise the applications cannot be installed. Therefore, we think that they are repackaged, and probably they are dangerous applications.

5. Conclusion and Future Work

In this paper, we conducted the research that tested the repackaged applications. In the experiment, we found that a total of 8 applications are not signed, 9 of the same 57 application have different hash values of fingerprints of each application. It means that it is possible to check repackaged applications using our approach. We also consider that it can be used in the malware analysis. In the future, we will research the malware analysis using the signed hash value of android application.

6. Acknowledgements

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No. 20120006492).

7. References

[1] Chung-Ying Huang, Yi-Ting Tsai, and Chung-Han Hsu. “Performance Evaluation on Permission-Based Detection for Android Malware,” In Proceeding of the Advances in Intelligent Systems & Applications, SIST 21, pages 111-120, 2013

[2] Yajin Zhou, and Xuxian Jiang. “Dissecting android malware : Characterization and evolution,” In Proceeding of the 33rd IEEE Symposium on Security and Privacy, pages 95-109, 2012.

[3] Blasing, Thomas, et al. "An android application sandbox system for suspicious software detection." Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on. IEEE, 2010.

[4] Google Play Store, http:/play.google.com/store/

[5] Unofficial Download Site, http://www.android-themes.com/, http://www.androidappsapkfree.com