**Abstract**

**1. Introduction**

//Android security

Android is a mobile operating system that is based on Linux kernel. Android’s open source nature, inherited from Linux, attracts many companies and developers’ interest, while this nature also brings some security threats. Last November, [The Financial Times](http://www.ft.com/cms/s/0/57d648e0-6404-11e4-8ade-00144feabdc0.html#axzz3Pgluvyc7) released a report that more than 40% of UK businesses were hit by mobile security breaches in last year. The mobile security is a heated subject currently.

//questions

In order to provide more security as well as greater functionality, the developers publish new versions of the apps in a period of time to keep their product alive. Whether the new version is always better than the old one? Whether the new version is worth downloading and installing? These are the questions we want to investigate in this research.

//app evolution

To answer these questions in terms of security, we take a look at application evolution. We compare the different versions of an application and use some security metrics to quantify the changed part. The metrics include the changed size, the changed count of re-delegation, the changed count of open components, the changed count of over-privileged permissions required, the changed count of dangerous permissions combination and so on. We propose a security analysis model that contains these metrics. From this model, we can understand the state of the art of Android’s application evolution. The tool we will build to compare the different versions can automatically detect the metrics we mentioned above and quantify them.

//some app evolution result

//update attack and result

When we observe the application evolution, we also pay attention to update attack. Update attack happens in the application upgrade period. Some malwares fake as safety software when users install them. However, the malicious codes exist in the update component. Once the users install them, they remind users a new version is available and induce users to update the application that contains the malicious codes. To take a deep look, we record the result of the malware using the same metrics that we used in studying application evolution, since we get an amount of results of changed parts between different versions, we compare these results to the malwares’ results. In order to make the results more reasonable, we use some attributes selection algorithms to remove some useless or less useful metrics. Then we try some classification algorithms, we find that when we use decision tree algorithms, we can get 93% accuracy to detect if the new version contains the malware samples.

\*(Permission combination using classification)

In summary, our contributions include:

1. We analyze Android application evolution in terms of security, the applications come from both official Google market and some third-party markets.

2. We develop a tool to compare the different versions of an application. The developers can use our tool to find if any security problems are missing in the latest version. And the users can use our tool to decide whether is worth installing the new version.

3. We can detect update attack using classification algorithms.

\*4. We compare the permissions used in legitimate applications and malware, and find some permissions combination used in malware are more frequent than those used in legitimate applications.

//thesis organization

The rest of this paper is organized as follows: Section 2 presents the android security mechanisms. Section 3 provides all metrics which are used to present an Android application. Then, section 4 shows the application evolution analysis of security. After that, section 5 illustrates the results of application evolution, followed by the detection of update attack in section 6. Finally, section7 is the related work and section 8 concludes this paper.

**2. Android Security Mechanisms**

**2.1 Sandbox**

//what

Application sandbox is a mean to isolate the applications from each other in Android system by assigning a user id and a set of permissions. It’s a kernel level security action that ensure the application has its own process space and memory which cannot modify by other if the other application doesn’t have the related permissions.

//how when

When the application installed in the device, it runs in its own sandbox and other applications cannot access and interfere. An application can only access its own files and files from other applications that explicitly showed this application has the access permission. If the applications are created by the same developers, the developers can make them share the same user id, then run in the same sandbox and share the resources in that sandbox.

**2.2 Permission**

//why

The application is isolation when running in the sandbox. However, when it wants to access some sensitive features which can only be accessed by the operating system, such as camera, location, telephony, network, Android provides permissions model to achieve this goal. Without this model, these sensitive features may be used by no applications or be used immoderately by any applications.

//what

Permissions mechanism is used to make some restriction when the applications want to access the sensitive API of the operating system.

// when how

An application can declare what permissions it needs in the manifest. Before the application installs in the device, the system will ask the users if they granted the respective permissions to this application. If the users agree to grant all requested permissions to the application, the installation continues, if not, the installation cancers. Unlike iOS, the user cannot choose which permissions they want to grant and which permissions they want to deny. However, the application can get the permissions not only from the user granting but also the application signing (we will talk about this later).

The permissions have four levels, normal permissions, dangerous permissions, signature permissions and signatureOrSystem permissions. Normal permissions can be granted automatically; dangerous permissions are inferred to those granted by the users; signature permissions are granted in the same sandbox, in the other words, signature permissions are granted to the same developers; SignatureOrSystem permissions are granted to pre-installed applications or the applications installed by the root.

\*\*\*\*user can turn off some functionality globally, what will happen when user uses that application which needs that function.

**2.3 Application Signing**

//what

Application signing is used to ensure the application security. It creates a certification between developers and their applications.

//how, when

Before placing an application into its sandbox, the application signing makes a relationship between user id and the application. The applications couldn’t be run on the Android without signing. With the same user id (running in the same sandbox), the applications can share the permissions which they required and can communicate with each other.

//why

By using application signing, it can be used in application update that it reduces some procedures. Since different versions of the same application can have the same certificate, the Package manager can verify this certificate. Then, the old version is replaced, the new version can have the permissions that the old one already had.

What’s more, the application signing can also ensure that an application cannot interrupt the other one unless they use ICC (we will talk about this later). But if the author is the same, the author can use the same application signing to make his/her applications to communicate.

**2.4 ICC (Inter-Component Communication)**

//what

Android application consists of components (activities, services, broadcasts and providers). Android platform provides a secure ICC that is similar to IPC (inter-process communication) in Unix system. The difference is that in ICC, the component as the receiver can be in the same application or in the other application.

//how

ICC is provided by binder mechanism which is in the middleware layer of Android. [Binder](https://source.android.com/devices/tech/security/overview/app-security.html) is a remote procedure call that is from a custom Linux driver. ICC is achieved by intents. An intent is a message that shows the target and/or some data. It can be used in the explicit communication if it identifies the receiver’s name, or used in the implicit communication that let the receiver to see if it can access this intent.

**\*\*\* Androguard, androwarn? need to write?**

**3. Metrics Selected**

When we consider presenting an Android application of security, we choose three main parts of metrics, 自身的属性？, vulnerabilities and sensitive APIs. 自身的属性？contain application’s size, classes number and components number. The Android security vulnerabilities, such as re-delegation, open components, dangerous permissions combination, hidden files and so on, are identified by many Android security research groups in these years. The sensitive APIs include the API related to network, location, contact information, SMS, phone calls and so on.

(There’s some overlapping between some vulnerabilities related to permissions and some APIs when we use them to present an application, for example, the over-privileged permissions requested may contains some sensitive APIs. However, in the later work, we use attribute selection algorithms to decrease the overlapping degree. )

3.1自身的属性？

1. Size

16. classes

17. components

\*\*\*18 permissions

2. Re-delegation

3. Open Components

4. Over-privileged permissions requested

5. dangerous permissions combination

6. hidden files

7. dangerous domains

8. root exploit

9. native codes, reflection, dynamic codes, crypto codes

10. audio, video

11. read network, network metered, read wifi

12. location lookup

13. contact access

14. sms access, send message, sms interception, disables incoming sms

15. make phone calls

**4. Security Analysis**

**5. App Evolution Result**

Example.

Open Component

有多少是有问题的

有多少remain the same, reg 和 send 分别是多少

有多少add, reg 和 send 分别是多少

有多少del, reg和send分别是多少

**6. Update Attack Detection**

**由于谷歌自身安全监测机制的保障，我们将从谷歌应用商店下载的应用看做是安全的。将从contagio下载的视为malware.**

**6.1 malware database**

**6.2 attributes selection**

**6.3 classification algorithms**

**7. Related Work**

**7.1 Android Security Extensions**

When we evaluate the application evolution, we select several metrics to present. These metrics are selected from the paper related to Android security.

Comdroid talks about if the intent doesn’t identify the recipient or the sender, it may be intercepted or suffer from spoofing attack. Open component metric is selected from this paper.

Permission re-delegation, also called the confused deputy attack, is when an application can pass its permission by making an API call to another application that doesn’t have this permission. XmanDroid[6], Quire[7], IPCInspection[8] list their methods to deal with this kind of problem.

Kirin[5] lists several dangerous permissions combination that can increase security risk. We use the results of Kirin, however, when we apply attributes selection algorithms to these metrics, it shows that the result is less useful.

Over-privileged permissions requested is also called privilege escalation. *Privilege Escalation Attacks on Android* explains this vulnerability and takes some actions to solve this problem.

**7.2 Android Malware**

*Malware Evolution* shows the change of Android malware in these years. The authors collect more than 1200 malware samples to analyze the evolution. *Mobile Malware Survey* shows that the incentives for writing malware include selling user information, stealing user credentials, making premium-rate calls and SMS and so on. Learning from these incentives, we record the API related to network, location, contact, SMS, phone calls as metrics.

**7.3 Update Attack**

*Malware Evolution*says that more recent Android malware families are adopting update attacks and drive-by downloads to infect users. Update attack is more tricky than common malware detection. Now the most popular way to detect malware is to use signature, which is not good to update attack detection. *Update attack* use network behavioral analysis to identify the update attack. However, we use data mining technique to detect the update attack, we compare the malware samples with the updated part.

**\*7.4 Dangerous permissions combination**

Kirin[5] lists several vulnerable permissions combination that can increase security risk. However, we use a different method to find new permissions combinations. We combine the permissions from malware samples. Then we use these possible combinations to distinguish the malware and legitimate applications. If the accuracy of the classification is high, it proves this combination is in effect.

**\*\*\* where hidden files, dangerous domains, root exploit, native code and so on is from?**

**8. Future Work**

**9. Conclusion**

**References**