Warning System for Detecting Malicious Applications on Android System

Sung-Hoon Lee and Seung-Hun Jin

***Abstract*—Android, Google open source operating system for mobile devices, has been rapidly grown. Android applications (apps) are uploaded to not only official market, play store, but also other alternative markets. These markets have no inspection system for detecting malicious apps. So, malicious apps can be easily uploaded on the market. Many studies have presented to detect malicious apps for the secure Android mobile environment. But without user participation, it is difficult to detect all of malicious apps.**

We present a Warning System for detecting malicious applications on Android system. The warning system monitors API calls related suspicious behaviors running on the service layer. After analysis malicious apps, we find that almost malicious apps use similar API calls. We proposed modified Android platform to warn suspicious API calls are running on the service layer. The warning system lets user realize the information of suspicious API calls and block the API calls.

***Index Terms*—Android security, malware detection, intrusion detection, API monitoring.**

I. Introduction

Since Apple's iPhone and Google's Android phone are released in 2009, the smart phone has been rapidly grown. Smart phones provide not only basic functionality such as calling service and SMS, but also additional functionality through different apps. Smart phone users download apps from the app market and use them. Apple operates only one app market which is App Store that includes an inspection system to check if the apps have malicious code or not. In contrast, Play Store, the official app market of Google, has no inspection system for checking apps. iPhone users can download apps from the only app store, but Android users can download apps through many ways such as Google play store, alternative markets, internet, Bluetooth, USB, and so on. Because of this, malicious apps of android can be easily distributed.

Malicious apps cause financial threats by collecting user information and sending premium-rate SMS messages. To protect your phone from the threats, many anti-malware companies have developed anti-malware apps. These anti-malware apps are normally based on the signature. The technic based on the signature is fast and simple to detect known malware. But it is not able to detect new malware and it takes 48days to receive a signature for a new malware [1].

Manuscript received November 10, 2012; revised January 24, 2013.

Sung-Hoon Lee is with the University of Science and Technology, Daejeon, Korea (e-mail: sunghoon1130@etri.re.kr).

Seung-Hun Jin is with Electronics and Telecommunications Research Institute, Daejeon, Korea (e-mail:jinsh@etri.re.kr).

In order to mitigate these threats from malware, many studies have been presented to detect malware on android platform [2]-[8]. But, it is still difficult to detect malware on android.

We present a warning system for detecting malware on Android system. The warning system monitors API calls related with suspicious activities and warns user with a message. An API is related to an Android framework. The Android framework is based on several libraries which are listed on the android developer site [9]. The warning system is able to detect for user suspicious activities running on the service layer without the user's consent and block the API calls. We find that malwares call specific API calls through analysis of android malware samples.

In summary, this paper makes the following contributions:

To understand how to run malicious code on Android and find specific API calls using suspicious activities, we have analyzed android malware samples and illustrated a structure of malwares.

We have proposed our system, warning system for detecting malicious applications, in android platform. In order to illustrate our point, we have constructed the system to monitor suspicious API calls running in the background of the system and block the API calls.

The rest of this paper is organized as follows: Section II introduces static analysis method for android malware including the malware sample collection. In Section III we analyze the malware we collected from the web. In Section VI we explain the warning system framework. In Section V we conclude and give possible future work.

II. Malware Analysis

In order to detect malicious activities of malware, we need to analyze malware. There are two methods which are static analysis and dynamic analysis. In this paper, we use static analysis.

A. Datasets

We collect Android malware from "Contagio mobile" [10] site that has several popular malware samples. Malware can be divided up into three types. (a). privilege escalation exploit for additional threats. (b). leakage of the user's personal sensitive data. (c). premium rate SMS billing. We focus on two types of malware samples, type (b) and type (c).

B. APK

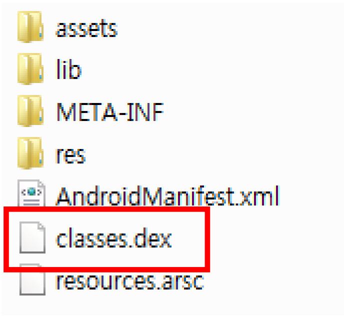
Android application package file (APK) is the file format used to distribute and install application. APK file consists of AndroidManifest.xml, Classes.dex, res directory, lib directory, META-INF directory, and resources. arsc as shown Table I [11].

TABLE I: THE ARCHITECTURE OF APK

File	Description
AndroidManifest.xml	XML file including information such as permission and description about application
Classes.dex	Binary execute file running on Dalvik Virtual Machine
/res	The directory including resource files such as icon, image, music and so on
/lib	The directory including the compiled code
/META-INF	The certificate of the application and the list of resources and SHA-1 digest and so on
resources.arsc	Compiled resource file

C. *Decompile*

The goal of decompile is to get an original java source codes. As said previous section, APK file is a compressed file. It can be extracted dex file by unzip tool. After extracting an apk file, it will be shown as Fig. 1.



assets
lib
META-INF
res
AndroidManifest.xml
classes.dex
resources.arsc

Fig. 1. Results after extracting apk file.

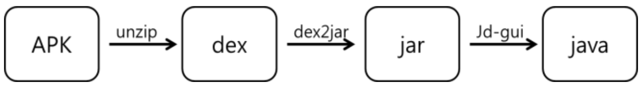


Fig. 2. Static analysis method step.

In order to get an original java source codes, we decompile classes.dex file through dex2jar [12]. A jar file aggregates java class files of APK. With jd-gui tool that displays java source codes of jar/class file [13], we can get original java source codes. Fig. 2 shows the entire step to get java source codes from apk file.

D. *Result of Static Analysis*

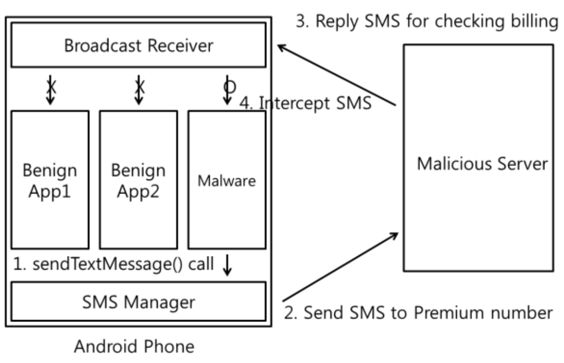
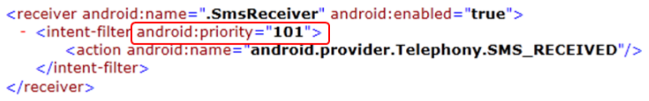


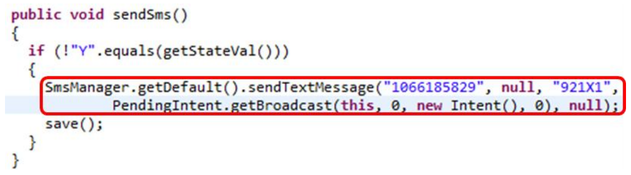
Fig. 3. Structure of premium rate SMS billing app.

We decompile several malwares and then analyze them. The structure of premium rate SMS billing malicious app such as Zsone [14] works as shown in Fig. 3. When malware is installed, malicious Broadcast Receiver is registered to broadcast messages from malicious server to only malware so that user can not realize whether specific number messages are delivered or not. Because the priority of malicious broadcast receiver is higher than SMS broadcast receiver as shown in Fig. 4. After malware is started, sending Text Message () of SMS Manager API on the service layer invoke to send a message with premium number which is already defined in the source code as shown Fig. 5.



```
<receiver android:name=".SmsReceiver" android:enabled="true">
  - <intent-filter android:priority="101">
       <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
    </intent-filter>
</receiver>
```

Fig. 4. Manifest.xml of malware.



```
public void sendSms()
{
  if (!"Y".equals(getStateVal()))
  {
    SmsManager.getDefault().sendTextMessage("1066185829", null, "921X1",
        PendingIntent.getBroadcast(this, 0, new Intent(), 0), null);
    save();
  }
}
```

Fig. 5. Premium number is already defined in the source code.

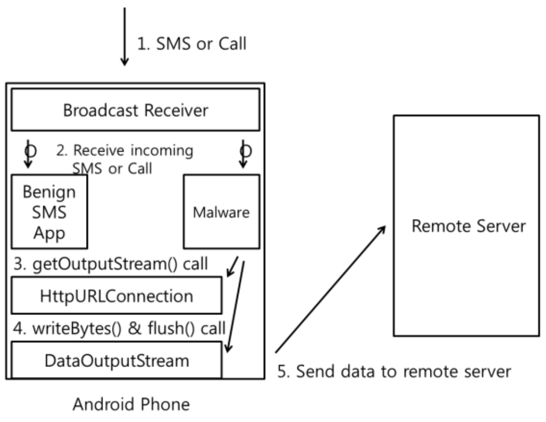

1. SMS or Call
Broadcast Receiver
2. Receive incoming SMS or Call
Benign SMS App
Malware
Remote Server
3. getOutputStream() call
HttpURLConnection
4. writeBytes() & flush() call
DataOutputStream
5. Send data to remote server
Android Phone

Fig. 6. Structure of malware related with leakage sensitive data.

Malicious Server receives a message from the user phone and then the server sent a message to the user phone for that SMS billing is charged. After the phone receives the message, malicious Broadcast Receiver which is already registered intercepts the message and the original SMS app does not receive the message. Almost of malware related with premium rate SMS billing use send Text Message ().

The structure of malware related with leakage user sensitive data works as shown in Fig. 6.

Like premium rate SMS billing malware, this malware also registers malicious broadcast receiver for broadcasting incoming messages or calls. The difference between previous malware and this malware is that benign app and malware receive information from the broadcast receiver at the same time. Malware creates a text file and logs incoming messages and calls into the file. And then malware periodically send the file with sensitive information such as IMEI, device ID, company to remote server.

As a result, malwares register malicious broadcast receiver and work on the service layer for hiding their malicious activities.

III. PROPOSED TECHNIC

We found that malicious activities using API calls work in the background of the system and the API calls are limited to be used for malware.

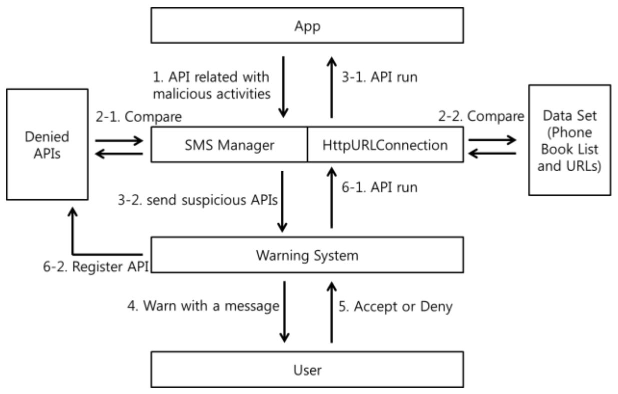


Fig. 7. Structure of proposed android platform.

Therefore, we present a warning system to monitor API calls related with malicious activities and to warn a user that the API calls work without the user's consent. For this system, we need to modify android platform as shown Fig. 7.

When the app runs, many API calls are working. If specific API calls such as send Text Message (), get Out put Stream (), write bytes (), and so on are calling by the app, each API calls compare with the dataset which stored phone number list from the user phone and URLs that is already registered by the benign app. If it exists on the dataset, the API calls run. If not, API calls are sent to the warning system and then the warning system pop-up a message with information related with API.

The user may accept or deny about the API. If the API is denied, the API is registered to Denied APIs file. Because malicious code runs periodically, even the user deny API, we need to create a file for denied API calls.

API calls is stored with PID. PID is a process ID and given for each apps as shown Fig. 8 [15]. So the system can classify whether this API is denied from the suspicious app.

The features of this system are as follows:

- It can monitor API calls running in the background of the system. In current android system, there is no way for user to realize which API calls are running in the background of the system. With the proposed system, the system warns a user that suspicious API calls are running in the background without the user's consent.
- It can block suspicious API calls if the user is not sure about running API calls. Once API calls denied, there's no additional operations. If the user wants to back, API calls is removed from the Denied API calls file.

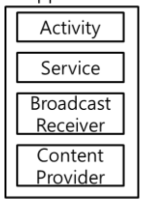


Fig. 8. App structure.

IV. Conclusion and Future Work

In this paper, we have analyzed two type malware which is sending premium rate SMS and is leaking user sensitive data to the remote server. The main reason to choose these malware is causing most financial threats. We have seen that android malware uses specific API calls for malicious activities. The premium rate SMS billing app uses send Text Message () of SMS Manager API with the target number which is already defined by malicious developer. And the leakage of user sensitive data app uses http communication API such as Http URL Connection () and URL Connection () for sending user data to remote server.

The existing anti-malware apps are based on signature for detecting malware. If malware is transformed to avoid detecting based on signature, the signature of malware is changed. Therefore the existing anti-malware apps do not detect new and unknown malware because they have no signature.

In order to address this problem, we have proposed a new framework to monitor and to block API calls related to suspicious app which is not scanned by the existing anti-malware apps.

There's a limitation that the user is familiar with android applications and has the minimum knowledge of malware. Because the user have a right to deny or accept warned API calls.

This work is simply the first step in a longer journey towards advanced detecting android malware. Next step is to compare normal apps to malicious apps for developing automatic algorithm system and to implement the warning system. In order to keep the latest dataset, it needs to periodically update normal API calls of benign apps. And the warning system sends the Denied APIs file to remote server for dynamic analysis.

References

[1] J. Oberheide, E. Cooke, and F. Jahanian, "Cloudav: Nversion antivirus in the network cloud," in *Proc. 17th USENIX Security Symposium*, 2008, pp. 2.2-1-2.2-6.

[2] W. Enck, M. Ongtang, and P. Mcdaniel, "On lightweight mobile phone application certification," in *Proc. 16th ACM Conference on Computer and Communications Security*, 2009, pp. 235-245.

[3] I. Burguere, U. Zurutuza, and S. Nadim-Tehrani, "Crowdroid: Behavior-Based malware detection system for Android," in *Proc. 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, 2011, pp. 15-26.

[4] W. Dong-Jie, M. Ching-Hao, W. Te-En, L. Hahn-Ming, and W. Kuo-Ping, "DroidMat: Android malware detection through manifest and API calls tracing," in *Proc. Seventh Asia Joint Conference on Information Security(Asia JCIS)*, 2012, pp. 62-69.

[5] T. Isohara, K. Takemori, and A. Kubota, "Kernel-Based behavior analysis for android malware detection," in *Proc. Seventh International Conference on Computational Intelligence and Security*, 2011, pp. 1011-1015.

[6] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," *IEEE Symposium on Security and Privacy*, pp. 95-109, 2012.

[7] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang, "Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets," in *Proc. 19th Network and Distributed System Security Symposium*, 2012.

[8] Y. Zhou, X. Zhang, X. Jiang, and P. Ning, "Droid MOSS: Detecting repackaged smartphone applications in third-party android marketplaces," in *Proc. 2nd ACM Conference on Data and Application Security and Privacy*, pp. 317-326, 2012.

[9] Google. Android API Reference. [Online]. Available: http://www.developer.android.com/reference/packages.html

[10] Mila. (November, 2012). Android Malware samples. [Online]. Available: http://www.contagiominidump.blogspot.kr/

[11] M. Blazek. (August, 2011). APK file format description. [Online]. Available: http://www.file-extensions.org/article/android-apk-file-format-description

[12] Pxb1988. (November, 2012). Dex2jar. [Online]. Available: http://www.code.google.com/p/dex2jar/

[13] E. Dupuy. (November, 2012). Jd-gui: Yet another fast java decompiler. [Online]. Available: http://www.java.decompiler.free.fr/?q=jdgui

[14] T. Strazzere. (May, 2011). Zsone Trojan malware in Android. [Online]. Available: https://www.blog.lookout.com/blog/2011/05/11/security-alert-zsone-trojan-found-in-android-market/

[15] A. Shabtai *et al*., "Google android: A comprehensive security assessment," *IEEE Security & Privacy*, vol. 8, no. 2, pp. 35-44, March 2010.

Sung-Hoon Lee received his B.S. Degree in Department of Computer Engineering from Chungju National University of Chungju, Korea in 2011. He is now pursuing a M.S. degree in Information Security engineering at University of Science and Technology, Daejeon, Korea. His research interests are android platform security and android malware detection

Seung-Hun Jin received his B.S. Degree in School of Computer Science and Engineering from Soongsil University, Seoul, Korea in 1993, the M.S. Degree in School of Computer Science and Engineering from Soongsil University, Seoul, Korea in 1995, and the Ph.D Degrees in Department of Computer Science and Engineering (Information Security) from Chungnam National University, Daejeon, Korea in 2005. He is currently a Team Leader/Principal Member of Engineering Staff of Authentication Research Team in ETRI (Electronics and Telecommunications Research Institute), Daejeon, Korea. His research interests include information security (PKI, authentication/authorization technique, privacy protection), mobile payment system, and computer/network security.