API and Permission-based Classification System for Android Malware Analysis

Jungsoo Park
Department of Software Convergence
Soongsil University
Seoul, Korea
ddukki86@ssu.ac.kr

Hojin Chun
School of Electronic Engineering
Soongsil University
Seoul, Korea
hjjjang4@naver.com

Souhwan Jung
School of Electronic Engineering
Soongsil University
Seoul, Korea
souhwanj@ssu.ac.kr

***Abstract*— APIs and permissions are often used as key features in static analysis process. In this paper, we classify applications into three categories according to their APIs and permissions: Benign, Suspicious, and Malicious. To achieve that, we define three levels of analysis. Level 1 has 19 categories like Network, System Summary etc., in a comprehensive meaning. Level 2 has 113 categories of detailed contents of Level 1 classification. In Level 3, not only does it match with the API's interface, class, or public method, but it also matches the permissions according to Level 2 classification. Based on this, API and permission based classification system were constructed as YARA Rule. The API, Class, and Public methods of each application are extracted from AndroidManifest.xml, classes.dex and matched with YARA Rule. We eventually raise user's awareness by providing insights about application behaviors, and let them judge whether to install the application on their devices.**

Keywords—Android; Malware; API; Permission; Claasification

Ⅰ. INTRODUCTION

The mobile OS Android, which Google first unveiled in 2008, has grown dramatically due to its open architecture, which has various advantages as an 'source open platform ', 'easy for application (app) development'. The company has a market share of 86% [1] in the global mobile OS market as of 1Q of 2017. However, these advantages make it easier for anyone to create and distribute malware. Because of this reason, new malicious APKs are emerging with fast pace. According to Gartner, about 750,000 new malware are generated in the first quarter of 2017 [2].

In this paper, malware analysis is classified into 3-level based on API and permissions. In Level 1, there are 19 categories of basic behaviors that can occur in malware. In Level 2, we divided the 19 categories into 113 categories. Based on this, we made it possible to distinguish between benign, suspicious and malicious. Level 3 distinguishes API, Class, Public Method, Permission, and String values based on the contents classified in Level 2. Through this distinction, we try to identify the behavior of malware in more detail. Also, it was converted to YARA rules and developed to be useful for malware analysis.

For APK analysis, API, Class, and Method analyzed in Permission analyzed in AndroidManifest.xml and classes.dex were used as input data and compared with classified data.

This paper introduces the related research such as API extraction and permission analysis in Chapter 2, and discusses the classification model proposed in Chapter 3. In Section 4, we introduce the results of YARA transformation based on the proposed model and conclude our work in Section 5.

Ⅱ. REALTED WORK

A. Extraction of API

The Android API is often used in static and dynamic analysis to determine the call flow to malware. Various Android malware analysis companies use commercial software such as dex2jar and apktool. After extracting classes.dex from APK, it extracts JAR file through commercial tool and converts to Java source code written by developer. However, this method does not work perfectly for the malicious APK analysis using the decompile analysis disruption technique, making effective analysis difficult. In a paper like FlowDroid [3], we propose a technique to analyze the syntax from source to sink, starting from the first line of the main method. It extracts from AndroidManifest.xml file, DEX file, and layout.xml file and analyzes it. However, FlowDroid has the disadvantage that obfuscated applications cannot be analyzed. To overcome these drawbacks, [4] introduces a technique to analyze classes.dex file by unzipping APK file. We analyzed the header of DEX file to analyze the offset value, field size of field such as string, type, and proto, and proposed a technique to parse class inheritance relation, internal field information, method information, etc. in class_defs area. In this paper, we have developed a program to extract and compare API based on the analysis of the classes.dex file proposed in this paper.

B. Permission Analysis in AndroidManifest.xml

All application root directories should have AndroidManifest.xml. The AndroidManifest.xml file contains essential information that an Android app needs to get to run the code [5]. This information includes the Android version code, version name, package name, minSDKversion, Activity name, and permissions. We use AXMLPrinter2.jar [6] to decode AndroidManifest.xml, and the file is changed by result.txt. Permissions are stored in the Manifest file in the following manner.

TABLE I. PERMISSION LIST

List	Subject
User-permission	Registering Permissions Required for Your Application
Permission	Registering permissions that an application allows for other applications
Permission-tree	Registering Permission Tree
Permission-Group	Registering Permission Group

C. *Analysis of malware by checking API and permission usage*

Various studies are being conducted to distinguish APIs and permissions that are commonly used in malicious APKs. Recently, they have been used as features of machine learning. Naser Peiravian et al. suggested 1,500 APK files were analyzed by using APKtool to analyze the AndroidManifest.xml file and used it as features of machine learning [7]. As shown in Fig. 1, the result of the usage of the frequently used permissions in the malicious APK is derived.

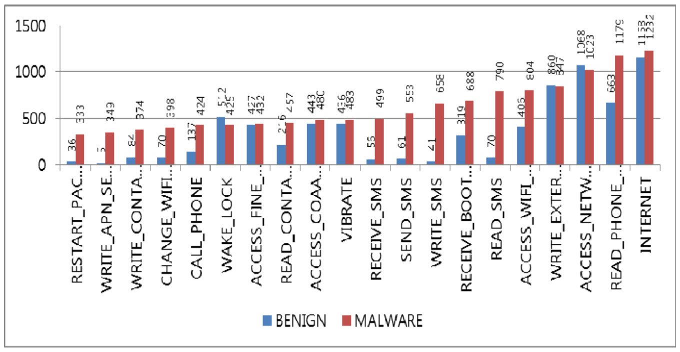


Fig. 1 Permission Usage

Shina Sheen et al. proposed a method to detect malware based on the permissions and API of malware [8]. Fig. 2 shows the frequency of use of the most frequently called APIs in malware and normal apps. In this paper, we propose a malware detection method using APIs and permissions that are characteristic of malware.

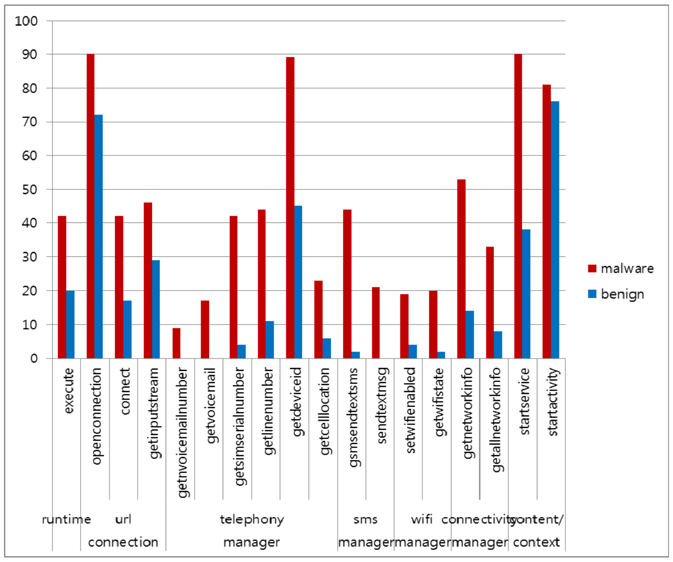


Fig. 2 API usage frequency

Ⅲ. API AND PERMISSION CLASSIFICATION

In this paper, we perform 3-level classification based on API and permissions. Level1 is a large category, such as Network, Sensitive Data Leakage, and System Summary, and is divided into 19 categories. Level 2 is the detail of Level 1 classification and classified into 113 categories. Level3 is about API and permission matching Level2 classification. The API has not only API but also Class, Interface, and Public Method. Based on this classification, YARA Rule was created so that API, class, and public method analyzed in classes.dex can be compared with permissions parsed in AndroidManifest.xml.

A. *Level 1 Classification*

In Level 1, classification of comprehensive meaning was performed as shown in Table 2. It is classified into 19 categories and classified based on contents mainly used in malicious APK. Especially, Anti Debugging, VM detection, etc. were tried to cope with the latest malicious APK by putting classification technique to cope with intelligent malware.

TABLE II. LEVEL 1 CLASSIFICATION

Level1 classification	Subject
AV Detection	Results from Virustotal
Change of System Appearance	Screen Lock, wallpaper change, etc.
Location Tracing	Locating with GPS
Operating System Destruction	Check system revision such as checking usage, deleting log
Spam, unwanted Advertisements and Ransom Demands	Confirmation of Ransomware feature through spam action and encryption using SMS and telephone
Privilege Escalation	Confirm elevation of access to administrator authority, etc.

Key, Mouse, Clipboard, Microphone and Screen Capturing	Checking the act of screen shot and recording through Audio, Media manager
E-Banking Fraud	Checking the behavior of accessing public certificate, accessing / proc file
Networking	Check all network activity such as internet connection, socket connection, bluetooth
Boot Survival	Actions related to Reboot
Stealing of Sensitive Information	Verification of accessing and transferring important information such as SMS data generation, authorized certificate access, contact access
Spreading	Check for WIFI changes, external storage access, etc.
System Summary	Actions to access dangerous permissions, verify sharing settings, etc.
Virtual Machine Detection	VM check by OS and CPU check
Anti Debugging	ADB connection, USB connection check
Malware Analysis System Evasion	Check for evasion technology such as X86, VM verification, / proc access
Hooking and other Techniques for Hiding and Protection	Encryption, backgroud, and running process check
Language, Device and Operating System Detection	System check of ISO Code, IEMI, MCC, MMC
HIPS / PFW / Operating System Protection Evasion	Check code injection technology with DEXClassLoder

B. Level 2 Classification

In Level 2, each category is classified based on the contents classified in Level 1, and added to the appendix. In the level 2 classification, the most frequently used benign apps are marked with ○, while the suspicious categories that can be used simultaneously in benign apps and malicious apps are classified as □, and those most commonly used in malicious apps are classified as △. Benign, suspicious, and malicious categories were categorized based on the results of related works, as well as on the android developer site based on risky permissions and permission groups. We also refer to the Joesecurity reports [9].

TABLE III. LEVEL 2 CLASSIFICATION

Level1	Level2	Classification
AV Detection	Antivirus detection for submitted file	△
Change of System Appearance	Acquires a wake lock	○
	Mutes ringtone sound	○
	Sets a new wallpaper	○
	May access the Android keyguard (lock screen)	□
	Sets a repeating alarm	○
Location Tracing	Queries the phones location (GPS)	□
Operating System Destruction	Deletes call logs/history	△
	Deletes other packages	△
	Kills background processes	△
	May wipe phone data	△
	Lists and deletes files in the same context	△
	Backup call	△
	Check the usage	△
Spam, unwanted Advertisements and Ransom Demands	Dials phone numbers	□
	May perform phone calls in the background	□
	May send SMS in background	□
	May use Google Cloud Messaging (GCM) or Google's Cloud to Device Messaging (C2DM) services	□
	May block phone calls / Accesses private ITelephony interface	□
	Sends SMS using SmsManager	□
	May dial phone number	□
	Has permission to perform phone calls in the background	□
	Has permissions to monitor, redirect and/or block calls	□
	Has permission to write to the SMS storage	△
Privilege Escalation	Tries to add a new device administrator	△
	Starts an activity on device admin enabled	△
	Checks if the device administrator is active	△
Key, Mouse, Clipboard, Microphone and Screen Capturing	Accesses the audio/media managers	○
	Has permission to record audio in the background	□
	Has permission to take photos	□
	Records audio/media	□
Networking	Downloads files from webservers via HTTP	○
	Performs DNS lookups	○
	Open an internet connection	○
	Urls found in memory or binary data	□
	Checks an internet connection is available	○
	Enables or disables WIFI	□
	Scans for WIFI networks	□
	Uses HTTP for connecting to the internet	□
	Uses HTTPS	○
	Found strings which match to known social media urls	○
	Tries to download files via HTTP but all files are no longer available	□
	Loads a webpage with cache disabled	□
	Uses bluethooth	○
Boot Survival	Has permission to execute code after phone	□

	reboot	
	Installs a new wake lock (to get activate on phone screen on)	○
	Starts/registers a service/receiver on phone boot (autostart)	□
	Executes code after phone reboot	□
Stealing of Sensitive Information	Creates SMS data (e.g. PDU)	□
	Has permission to read contacts	□
	Has permission to read the SMS storage	□
	Has permission to read the call log	□
	Checks if a SIM card is installed	□
	Has permission to read the default browser history	□
	Has permission to read the phones state (phone number, device IDs, active call ect.)	□
	Has permission to receive SMS in the background	□
	May take a camera picture	△
	Queries MMS data	□
	Has permission to query the current location	□
	Has permissions to create, read or change account settings (inlcuding account password settings)	□
	Queries email messages	□
	Queries media storage location field	□
	Queries stored mail and application accounts (e.g. Gmail or Whatsup)	□
	Queries the Googlemail Account Name	□
	Monitors incoming Phone calls	△
	Monitors incoming SMS	△
	Queries a list of installed applications	□
	Queries camera information	□
	Queries phone contact information	□
	Queries stored mail and application accounts (e.g. Gmail or Whatsup)	□
	Monitors outgoing Phone calls	△
	Creates SMS data (e.g. PDU)	□
	Parses SMS data (e.g. originating address)	□
	Queries SMS data	□
	Uses reflection	□
	Accesses Class Loader via Reflection	□
Spreading	Accesses external storage location	□
	Has permission to change the WIFI configuration including connecting and disconnecting	□
	Scans the access points for available WIFI networks	□
System Summary	Loads native libraries	○
	Reads shares settings	□
	Requests potentially dangerous permissions	△
Virtual Machine Detection	Accesses android OS build fields	□
Anti Debugging	Access the class loader (often done to load a new code)	□
	Creates a new dex file (likely to load a new code)	□
	Creates a new jar file (likely to load a new code)	□
Malware Analysis System Evasion	Accesses android OS build fields	□
	Queries several sensitive phone informations	□
	Queries the unique operating settings	□
	Accesses /proc	□
	Tries to detect Android x86	□
	Tries to detect Virtualbox	□
Hooking and other Techniques for Hiding and Protection	Uses Crypto APIs	□
	Has permission to terminate background processes of other applications	□
	Has permissions to monitor, redirect and/or block calls	□
	Aborts a broadcast event (this is often done to hide phone events such as incoming SMS)	□
	Queries list of running processes/tasks	□
	Has permission to draw over other applications or user interfaces	□
	Has permission to query the list of currently running applications	□
	Removes its application launcher (likely to stay hidden)	△
HIPS / PFW / Operating System Protection Evasion	Uses the DexClassLoader (often used for code injection)	□

C. *Level 3 Classification*

Level3 is based on the contents classified in Level2, API, Class, and Public Method which are used in actual Android. Interface Class [10]. An example is shown in Table 3 below. It refers to the developer site where the Android API List is defined.

TABLE IV. LEVEL 3 CLASSIFICATION EXAMPLE

Level1	Level2	Level3
Hooking and other Techniques for Hiding and Protection:	Queries list of running processes /tasks	android.app.ActivityManager.getRunningAppProcesses android.app.ActivityManager.getRunningTasks android.app.ActivityManager.RunningAppProcessInfo android.app.ActivityManager.RunningServiceInfo android.app.ActivityManager.RunningTaskInfo android.app.ActivityManager.TaskDescription

Also, in the case of the first AV Detection of classification, Virustotal API[11] was used to receive and immediately reflect Virustotal's score. AV Detection scores can be quickly and accurately reported to users by using the Virustotal API, which provides analysis results to a large number of companies without having a variety of signatures.

IV. Confirmation of analysis result using YARA RULE

YARA is a tool used to identify and classify types of malware to analyze malware [12]. You can classify malware based on text or binary information contained in the sample. YARA determines the rule logic by description, name, rule, set of strings, Boolean expression. It can be applied to a file or a running process to determine if the rule belongs to the type of malware described. The rules of YARA are used in similar syntax with the way C declares structures for easy understanding and writing.

In this paper, the 3-level classification system is summarized and developed to be compared with the analyzed .txt file by changing it to YARA. Fig. 3 shows the operation process for the analysis, and Fig. 4 shows an example of the created YARA.

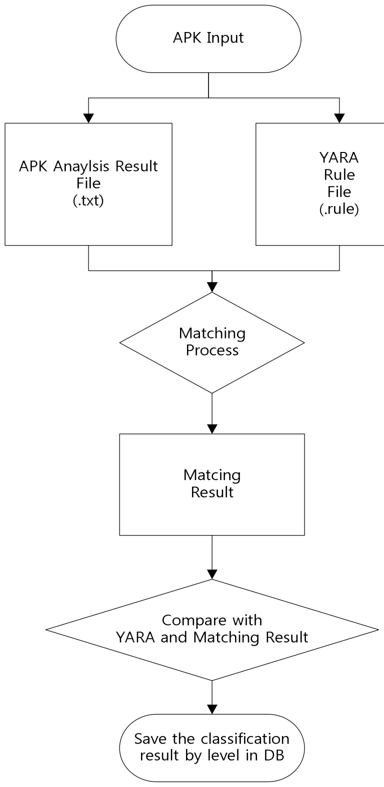
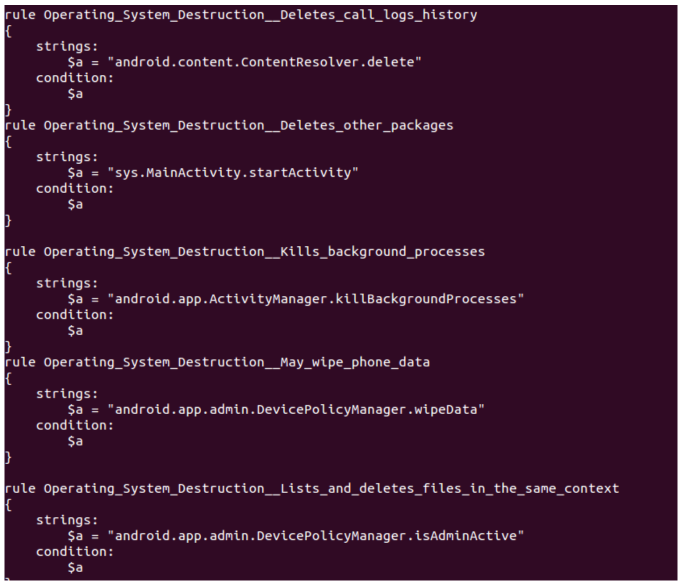

APK Input
APK Anaylsis Result File (.txt)
YARA Rule File (.rule)
Matching Process
Matcing Result
Compare with YARA and Matching Result
Save the classification result by level in DB

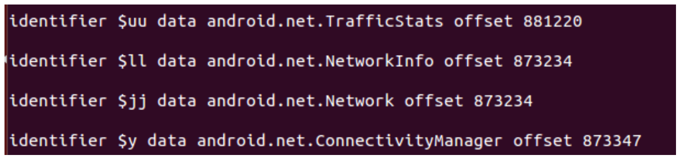
Fig. 3 Operation Flow



```
rule Operating_System_Destruction__Deletes_call_logs_history
{
    strings:
        $a = "android.content.ContentResolver.delete"
    condition:
        $a
}
rule Operating_System_Destruction__Deletes_other_packages
{
    strings:
        $a = "sys.MainActivity.startActivity"
    condition:
        $a
}
rule Operating_System_Destruction__Kills_background_processes
{
    strings:
        $a = "android.app.ActivityManager.killBackgroundProcesses"
    condition:
        $a
}
rule Operating_System_Destruction__May_wipe_phone_data
{
    strings:
        $a = "android.app.admin.DevicePolicyManager.wipeData"
    condition:
        $a
}
rule Operating_System_Destruction__Lists_and_deletes_files_in_the_same_context
{
    strings:
        $a = "android.app.admin.DevicePolicyManager.isAdminActive"
    condition:
        $a
```

Fig. 4 YARA Rule creation file

YARA starts with the keyword rule, which follows the rule of the identifier. Level 2 classification is followed by Level 1 classification, followed by Level 2 classification, based on two underscores after rule. The generated YARA is processed as a String match with the txt file containing the analysis result, and the matching result is shown in Fig 5.



```
identifier $uu data android.net.TrafficStats offset 881220
identifier $ll data android.net.NetworkInfo offset 873234
identifier $jj data android.net.Network offset 873234
identifier $y data android.net.ConnectivityManager offset 873347
```

Fig. 5 Matching result

In the matching result, the values in txt file and YARA are classified as data area. After that, the value of the data area is checked again by YARA and the Level1 and Level2 values are confirmed and stored in the DB. The analyst can judge whether the APK is malicious based on the results stored in the DB.

V. Conclusion

In many existing papers propose an analysis method of malware by distinguishing APIs and permissions which are frequently used in malware. In this paper, we additionally classify malware classification system using API and permissions into 3-level for Android malware analysis and convert it into YARA type and compare it with AndroidManifest.xml file and results extracted from classes.dex file Method. If the classification system is used, analysts can acquire threat information more quickly, which can help improve malware detection rate. In addition, APIs for technologies such as anti-debugging, anti-emulation, and root detection are summarized to make it possible to cope with recent intelligent malicious codes. It is expected that more accurate results will be obtained if malware score is calculated or used for machine learning later on.

ACKNOWLEDGMENT

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2017-0-01853, Machine Learning based Intelligent Malware Analysis Platform) And This work was supported by the IT R&D program of ATC under the MOTIE/KEIT. [10045904, The development of Fundamental Technology for Security as a Service (SecaaS) Framework under cloud computing environment and the implementation of 1Gbps mobile data loss prevention (DLP) service based on the SecaaS Framework.]

REFERENCES

[1] Tam, Kimberly, et al. "The evolution of android malware and android analysis techniques," ACM Computing Surveys, 2017

[2] Zhu, Dali, et al. "DeepFlow: Deep learning-based malware detection by mining Android application for abnormal usage of sensitive data," Computers and Communications (ISCC), 2017 IEEE Symposium on. IEEE, 2017.

[3] Arzt, Steven, et al. "Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps," Acm Sigplan Notices 49.6, 2014.

[4] Sangmin Ha. “A Study on Android Malware Flow Analysis based on Dalvik Instruction,” (Master’s thesis), Available from RISS, http://www.riss.kr, 2017.

[5] Peiravian, Naser, and Xingquan Zhu. "Machine learning for android malware detection using permission and api calls." Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on. IEEE, 2013.

[6] AXMLPrinter2.jar, https://code.google.com/p/android4me/downloads/detail?name=AXMLPrinter2.jar

[7] Peiravian, Naser, and Xingquan Zhu. "Machine learning for android malware detection using permission and api calls," Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on. IEEE, 2013.

[8] Sheen, S., Anitha, R., Natarajan, V. "Android based malware detection using a multifeature collaborative decision fusion approach, " Neurocomputing 151, 905–912, 2015.

[9] Joesecurity, https://www.joesecurity.org/joe-sandbox-reports#android

[10] Android API, https://developer.android.com/reference/packages.html

[11] VirusTotal, https://www.virustotal.com.

[12] Yu, Rowland. "Android packers: facing the challenges, building solutions," Proceedings of the Virus Bulletin Conference (VB'14). 2014.