

Malware Executables Analysis Using Hybrid Analysis Technique for Android Devices in Sandbox

Aman

Dept. of Computer Science & Engineering
Central University of Punjab, Bathinda, Punjab, India

Gunseerat Kaur

Dept. of Computer Science & Engineering
Lovely Professional University, Phagwara, Punjab, India
Gunseerat.22157@lpu.co.in

Abstract: Patronizing the evident increase in hand held device usage, cyber criminals have extended their spiteful doings towards Android operating system. Malware researchers are reporting frightening rise of detected Android malware from 2015 to 2018. In order to have some control over 500 new Android applications that are being released every day, there is need for a analysis technique that can give better results and can quickly detect malware instances in applications which earlier used techniques i.e. static and dynamic analysis technique lacked. In classification of static analysis of features and metrics are deduced from Androidmanifest.xml while in dynamic analysis of the system the metrics are derived from system logs and code packages possible collected when the system is in runtime. In this paper an effective method of deriving features using Vx Sandbox are analyzed on varied set of statistics. Analysis Platform is an automated framework for hybrid analysis of Android applications that combines aspects of both the techniques, i.e. combining dynamic and static analysis of prevalent malicious activity

Keywords: *Malware, Sandbox, Metamorphism, Android, API calls*

I. INTRODUCTION

There is a thin line of difference between mobile devices, laptops and other devices, which has emerged in last decade like tablet PCs as all of these devices are connected with each other and can communicate. Due to these devices being connected with internet, these are exposed to gain sensitive information about the users[1]. With devices having all these features, need for security of these smart phones is also felt. The hand-held devices have operating systems ranging from Android, palms and embedded Linux etc[2]. As a mainstream market share holder, majority of smart phones are being manufactured with Android OS. Declaring an openness in the system, with bulk loads of apps catering to users. This turns in as another set of major problem for such an operating system. The number of the Android devices are increasing day by day, concern of user security is becoming the main issue. Android is a very popular platform for hackers, because vulnerable users can be deceived by similar genre of apps and any installation or giving permissions could compromise the users' data[3]. Most of the times, even popular applications are also able to leak user's sensitive information and share it to the hacker or attacker without their consent.

This very reason caters to the cyber-criminals who conveniently make use of loops holes and exploit by breach of privacy. A leading anti-virus manufacturer claims that in the very first quarter of 2018[4], using SOCKS proxy and HTTP proxy varied number of backdoor attacks and threats have surfaced. The reports state that the deception based attacks have risen as the popularity of the apps play a very major role. The higher the popularity, the likelier is encouragement of other users to download and try[5]. These convincingly fake paradigms increase the risk, and this increase was majorly targeted towards the Android platform. This paper focuses on threats prevalent to Android devices and hybrid techniques for detecting the malware and its effectiveness as compared to static and dynamic analysis.

The paper is organized as follows: Section II discuss the overview of Android OS, section III discuss about Android applications, section IV discuss the security threats in android devices, and section V/VI presents the classification of malwares and their infection methods. Section VII illustrates the detection process of malwares by explaining different detection and analysis techniques that is followed by section VIII which discusses research methodology, whereas section IX presents the analysis of results. Section X outlines the conclusion and future scope.

II. OVERVIEW OF ANDROID OS

The android stack has layers of different programming languages used throughout its implementation hierarchy, in figure 1 it can be seen how different layers of architecture in android are comprising of various set of languages, this makes various types of attacks prevalent to each implementation.

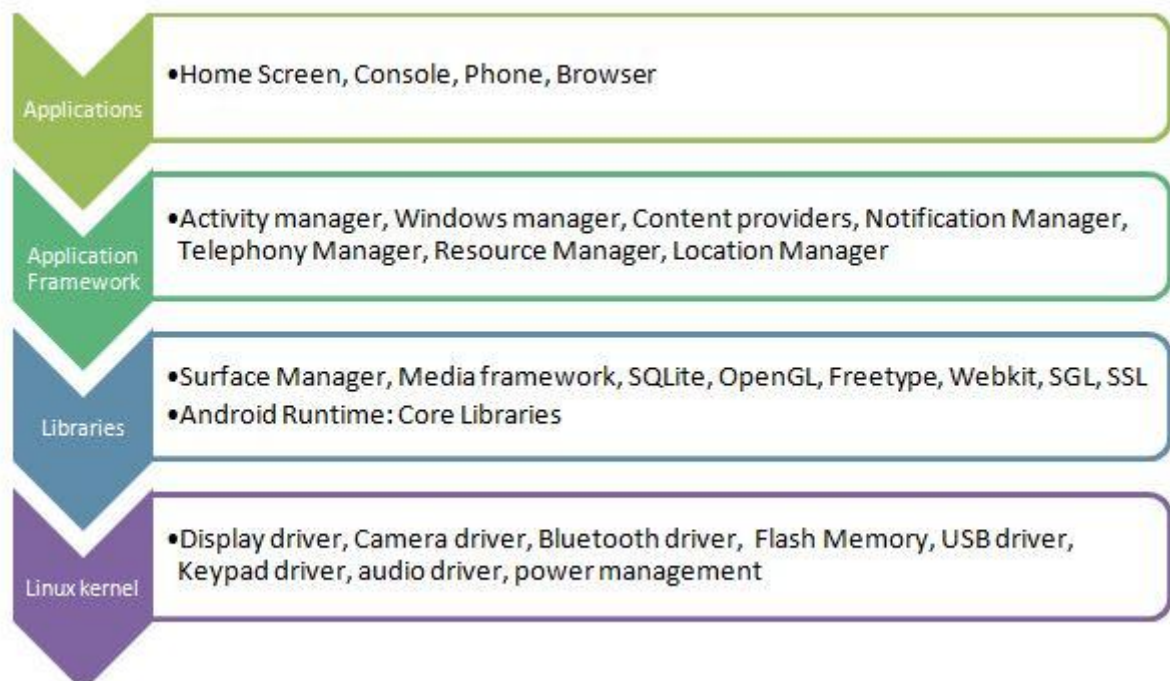


Figure 1. Android architecture

- Drivers like USB, IPC, power management system, audio drivers, camera drivers are allocated under the Linux kernel and play an important part in the hardware management.
- The libraries comprise of upper connections to database, SSL[6], SGL, core and media management. This establishes a functionality with core libraries.
- During runtime the android compiles the applications with installation.

- The final layer which interacts with the user and works with developed applications in library and packages. The layers in this component are:
 - Activity manager: Interacts with applications with user interface
 - Intent/ Notification Manager: Establishes inter-process communication between components
 - Content Manager: Establishes process to process data passing.
 - Telephony Manager: Attains the identities on hardware, for instance IMEI, SIM serial numbers.

III. ANDROID APPLICATIONS

The android applications coagulate on the top-most layer, provisioning an interaction between users and devices. The original form of applications are in .apk format, the lower category of packages are assigned in .ZIP format which contains .dex files, external libraries and AndroidManifest.xml file describing applications[7]. The permissions are composed in manifest file along with services, broadcast receivers, file permissions and other hardware requirements such as minimum API versions. The flow of information is maintained between various channels and layers of android architecture[2]. A particular service will invoke a particular call and achieve a paradigm from a transmitted request to set library calls. The calls are then forwarded to varied system objects and establish the communication between kernel and hardware.

IV. SECURITY THREATS

Application based threats have shown an increase in abundance due to openness of the system, the apps categorically targets users by studying their behavior and implementing a sophisticated attack therewith. The data is often collected from public domains and this helps attacker understand the users capability[3]. Often the attackers make use of phishing and trick the naive users into visiting webpages or apps of counterfeit to gain information[8]. The major reason for such approach is:

- Wide market share is held by Android operating system, which is open source system and profiled into a more prone category.
- Ease of hosting any developed content online, makes it vulnerable for attackers to gain information



Figure 2 Basic family of attacks on android

V. CLASSIFICATION OF ATTACKS

There exists various attacks in the wild that can compromise the security of Android smartphones. Three main categories of attacks are as follow:

i **Malware**- Malware is a program or file that is harmful to a computers or mobile devices. These malicious programs can steal, encrypt or delete the user's sensitive data; or can hijack core-computing functions without user's knowledge[9].

Few of the malware attacks are as follow:

- **Bluetooth attacks**: With this attack, contacts, messages, photos are stolen from the user's phone without his knowledge. This attack happens when user's has left his Bluetooth on by mistake[10].
- **SMS attacks**: This is most common attack happening most of the times. In this attack, users are annoyed with the phishing links or advertising messages.
- **GPS attacks**: In this attack, user's location is tracked constantly for various purposes, e.g. advertising or spying.
- **Phone Bootloader Unlocking**: With a locked bootloader on Android devices, it is virtually impossible to flash a Custom ROM and forced attempts void warranty as well as usually end up in bricks. Therefore, the first step is to always unlock the bootloader to flash the malicious applications.
- **Premium rate attacks**: In this, premium rate messages are sent from the user's phone i.e. International SMS with his knowledge and this costs him a lot.
- **Grayware**: Grayware refers to applications that have annoying, undesirable, or undisclosed behavior but do not fall into any of the major threat categories[11].

ii **Spyware**: Spyware are the application, which are installed on a user's device secretly to collect personal information or monitor his activities.

VI. MALWARE INFECTION METHODS

There exists many methods which can infect the smartphones with malware. The following are four different methods by which malware can be installed on the phone:

a) Repackaging of the legitimate application

A very common method devised by attackers is to create a package similar looking to a legit or a common named app. These repackaged applications are then advertised on the Google play store and given more ratings[8], once a user installs them giving permissions it triggers a malicious payload reverting information to its source. User's device could be vulnerable by being downloading and installing these repackaged applications. In a research that was carried over for two years, it was found that 86% repackaged legitimate application includes malicious payloads[12].



Figure 2. Repackaging legitimate app

b) Exploiting Bug in Android's application

There could be some other form of worms or Trojans linked through the binaries or libraries of apps. These could lead to compromise user's data and personal information.

c) Fake applications

Misnomer or look alike-applications are often uploaded on the play store which can in turn deceive users to install and get phished by the application creators. Once taken permissions these applications can compromise the users' device and cultivate it as an information source[10], [13].

d) Remote Installation

Another scenario is gaining user's credentials and installing applications through a remote interface. Every time the user comes in the range of internet, it will install and gain information from the device, worsening the situation further for user.

VII. MALWARE DETECTION AND ANALYSIS TECHNIQUES

Given the enormous growth of Android malware, lots of applications has to be analyzed by the security researchers in a limited period of time to understand the purpose of the application and then to develop the techniques for countermeasure. When it comes to application analysis for the malware, traditional penetration testing is not enough, and analysis of the source code using reverse engineering only tell a part of the story which is also not enough. Static analysis involves manual reviews that is a complex task and there are static source code analyzers like IDEs (Integrated development environment) also, which can look at some or all the code but it takes much time[14]. Analyzing the applications with static analysis technique can reveal plentiful of the issues, but you never know how exactly particular application is vulnerable, and some of the aspects are always missed during the analysis.

Dynamic analysis helps to solve this problem by compiling and/or executing and analyzing the code in real time. But performing both static and dynamic analysis together can give some significant result. If both the techniques are used standalone, no desired result is obtained which can tell actual problem with the application. Hybrid security analysis combines the aspects of the techniques, i.e. static analysis and dynamic analysis[8]; and this can help. The real benefit in taking a hybrid approach is that it will allow an analyst to find out security flaws, which never came out with any of the analysis techniques before and will matter in unique situations[2]. With this technique, understanding of the tangible source code and flaws at that level will provide better insight into the application manipulation in real-world setting, across all possible attack surfaces.

VIII. METHODOLOGY

Hybrid analysis technique, which combines both static analysis and dynamic analysis during application runtime, can help finding the issues in application that any of the previous technique could not find standalone.

Methodology:

- i. Prepare the simulation environment as shown in TABLE I.
- ii. Prepare the data set of various applications from different sources so that these applications can be analyzed for malicious patterns (TABLE II).
- iii. Selection of the sandbox.

VxStream Sandbox is chosen for research that is a high-end malware analysis framework with a very nimble architecture. It can process thousands of files automatically as a web service when implemented on a very large scale. It detects and analyzes unknown threat using unique hybrid technology. VxStream Sandbox includes the web service, API, runtime monitors, load balancing controller, hybrid analysis technology, report generator, all indicators, signatures, etc.

- iv. Simulate the applications on VxStream Sandbox by uploading them. Figure 3 shows all the options available.

Figure. 3. Advanced options during analysis

- v. Analyze for the malicious patterns e.g. Malicious indicators, Suspicious indicators, etc. (TABLE I).

Table I. Simulation Environment

No. of Applications	Approach Used	Parameters to be checked
40	Hybrid Analysis using VxStream Sandbox	Malicious Indicators Suspicious Indicators Potential URLs File Permissions File Activities

Parameters to be checked:

In Table I, these parameters provide vulnerability in the application. Here is why these parameters are checked:

- **Malicious Indicators:** Patterns in Android application due to which application can be categorized under malicious. E.g. Events which are started to leak sensitive information, malicious strings match, code injection, etc[15], [16].

–

- **Suspicious Indicators:** Patterns in Android application due to which application can be categorized under suspicious. E.g. Suspicious permissions, IP addresses, Sensitive API calls, etc.
-
- **Potential URLs:** URLs present in application which are either adware or phishing urls.
-
- **File Permissions:** Permission access in application which can access files present in device or can access phone state.
-
- **File Activities:** Activities triggered with file permissions.

In Table II, a list of 40 Android applications which is taken as dataset is shown. The list is obtained by examining the antivirus security announcements, threat reports from various mobile antivirus companies carefully and by looking at the permissions accessed by the app. For each android application, which was suspicious to have malware based on the permissions is taken in application dataset. The number of samples in dataset collection are taken from different sources, i.e., either from *Google Play Store* or from alternative Android Markets.

Table II. Android Applications Dataset

<i>Application Name</i>	<i>Official Market</i>	<i>Alternate Market</i>
Fake Call	✓	✓
Hair Care Tips	✓	✓
Payson UT Energy Conservation	✓	✓
Accounting Ratio Calculator	✓	✓
Humphrey Solitaire	✓	✓
PIP Lock Screen Passcode	✓	✓
Mobile Summarizr	✓	✓
Traveline Scotland	✓	✓
GO SMS Pro	✓	✓
Fish Adventure Aquarium	✓	✓
Cargo Ship Car Transporter 3D	✓	✓
GO Keyboard Pink Hearts Theme	✓	✓
SAMASTHA Directory	✓	✓
Love or Friendship Calculator	✓	✓
Fingerprint Age Detector Prank	✓	✓
UnBlock The Block-Puzzle Game	✓	✓
Runtastic Push-Ups PRO Trainer	✓	✓

GBWhatsapp PLUS	✓	×
AIO File Manager	✓	✓
신한S뱅크 mini	✓	✓
Simple Notepad	✓	✓
XPOSED IMEI Changer	✓	✓
Kaon Biz Apps	✓	✓
Mermaid kiss 2015	✓	✓
Christmas Live Wallpaper Free	✓	✓
Alarm Anti Theft Screen Lock	✓	✓
Atlantis Diamonds	✓	✓
Password Terminator	✓	✓
Warrior Rush	✓	✓
Uber	✓	✓
Cartoon Photo Frames for Kids	✓	✓
Spotify Music	✓	×
CamScanner -Phone PDF Creator	✓	✓
Messenger	✓	✓
Free to Read Detective Novels	✓	✓
Block Puzzle Quest	✓	✓
Space Calculator	✓	✓
GoingDutch	✓	✓
Camera360	✓	✓
Gaukeen	✓	×

IX. ANALYSIS OF RESULTS

In this section, once the application data is gone through the hybrid analysis technique, a result table containing all the parameters is created which needed to be checked for the suspicious and malicious behaviour (APPENDIX A.1).

Table A.I shows the malicious indicators, suspicious indicators, potential URLs, file permissions, file activities and extracted files from each application. Malicious behavior indicates the malicious patterns in the application and suspicious indicators indicate that app is not fully malicious but it's suspicious to have malware or intent to do some malicious activity.

From the analysis table, eight applications come under suspicious category, i.e. Hair Care Tips, Humphy Solitaire, Travel Scotland, Go Pro SMS, GBWhatsApp, Xposed IMEI Changer, Cartoon Photoframe for Kids, Block Puzzle Quest and five applications come under

malicious category, i.e. Cargo Ship Car Transporter 3D, Unblock the Block – Puzzle Game, AIO File Manager, S Mini Bank, and Camera 360.

X. CONCLUSION

The main aim of this research was to evaluate the effectiveness of combining the static analysis and dynamic analysis techniques for detecting the malware in Android applications. It's observed that none of the techniques – static and dynamic analysis is effective standalone. Static analysis gave five applications as malicious application, i.e. AIO File Manager, Atlantis Diamonds, Password Terminator, Warrior Rush and Camera 360; whereas

dynamic analysis gave three applications as malicious, i.e. AIO FILE MANAGER, XPOSED IMEI CHANGER, GO SMS PRO which is less in number but results obtained by dynamic analysis is more accurate as it took deeper look into the application run time behavior.

In this paper, hybrid technique using VxStream sandbox is used, which combines the aspect of both static and dynamic analysis technique. Hybrid analysis technique gives eight applications as suspicious applications and five applications as malicious, i.e. Cargo Ship Car Transporter 3D, Unblock the Block – Puzzle Game, AIO File Manager, S Mini Bank, and Camera 360, which is same in number as find out by static analysis technique, but the accuracy is better in Hybrid analysis. Hybrid analysis detects four applications more malicious than static analysis and two more than dynamic analysis.

In static analysis technique, all the results were based on hit and trail; and results accuracy was main question. Then in dynamic analysis, all the results were based on run time behavior of the application, and malware maybe set to run at some specified condition or time. But here, when the same applications data set was analyzed with hybrid analysis, it gave confirmed result categorizing the application under suspicious and malicious categories. Additionally, the combination of both these analysis techniques have proven to be very effective than the individual ones.

XI. REFERENCES

- [1] G. Robinson and G. R. S. Weir, "Understanding android security," *Commun. Comput. Inf. Sci.*, vol. 534, pp. 189–199, 2015.
- [2] M. Spreitzenbarth, F. Freiling, F. Echter, T. Schreck, and J. Hoffmann, "Having a Deeper Look into Android Applications," *Symp. Appl. Comput.*, pp. 1808–1815, 2013.
- [3] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, and S. Dolev, "Google Android: A State-of-the-Art Review of Security Mechanisms," 2013.
- [4] McAfee, "McAfee Labs Threats Report: December 2018," *McAfee Labs Threat. Rep.*, vol. 2019, no. 1, p. 4, 2019.
- [5] S. Y. Yerima, "Longitudinal performance analysis of machine learning based Android malware detectors," no. May, 2019.
- [6] G. Kaur and S. Singh, "Comparison of Machine learning algorithms in Anomaly detection.," *Int. J. Adv. Res. ...*, vol. 8, no. 5, pp. 2241–2247, 2017.
- [7] Y. Zeng, K. G. Shin, and X. Hu, "Design of SMS commanded-and-controlled and P2P-structured

- mobile botnets,” in *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks - WISEC '12*, 2012, p. 137.
- [8] P. Faruki *et al.*, “Android security: A survey of issues, malware penetration, and defenses,” *IEEE Commun. Surv. Tutorials*, vol. 17, no. 2, pp. 998–1022, 2015.
- [9] A. I. Journal *et al.*, “MAMA: MANIFEST ANALYSIS FOR MALWARE DETECTION IN ANDROID MAMA: Manifest Analysis for Malware Detection in Android,” *Cybern. Syst.*, vol. 44, no. November, pp. 469–488, 2016.
- [10] X. Chen *et al.*, “Android HIV: A Study of Repackaging Malware for Evading Machine-Learning Detection,” *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 987–1001, 2020.
- [11] E. Viegas, A. Santin, A. França, R. Jasinski, V. Pedroni, and L. Oliveira, “Towards an Energy-Efficient Anomaly-Based Intrusion Detection Engine for Embedded Systems,” *IEEE Trans. Comput.*, vol. 66, no. 1, pp. 1–14, Jan. 2016.
- [12] A. Demontis *et al.*, “Yes, Machine Learning Can Be More Secure! A Case Study on Android Malware Detection,” *IEEE Trans. Dependable Secur. Comput.*, vol. 16, no. 4, pp. 711–724, 2019.
- [13] M. Hicham, N. Abghour, and M. Ouzzif, “Device-To-Device (D2D) Communication Under LTE-Advanced Networks,” *Int. J. Wirel. Mob. Networks*, vol. 8, no. 1, pp. 11–22, 2016.
- [14] R. Johnson, Z. Wang, C. Gagnon, and A. Stavrou, “Analysis of android applications’ permissions,” *Proc. 2012 IEEE 6th Int. Conf. Softw. Secur. Reliab. Companion, SERE-C 2012*, no. November 2011, pp. 45–46, 2012.
- [15] T. Bläsing, L. Batyuk, A. D. Schmidt, S. A. Camtepe, and S. Albayrak, “An android application sandbox system for suspicious software detection,” *Proc. 5th IEEE Int. Conf. Malicious Unwanted Software, Malware 2010*, pp. 55–62, 2010.
- [16] J. M. San Miguel, M. E. M. Kline, R. A. Hallman, S. M. Slayback, A. Rogers, and S. S. F. Chang, “Aggregated Machine Learning on Indicators of Compromise in Android Devices,” pp. 2279–2281, 2018.

XII. APPENDIX A

Table A.I. Result of Hybrid Analysis using VxStream Sandbox

S. No.	Malicious Indicators	Suspicious Indicators	Potential URLs	File Permissions	File Activities	Extracted Files	Result
1.	TelephonyManager.getLine1Number .getDeviceId SMSManager.sendMultipartTextMessage	INTERNET READ/WRITE_EXTERNAL_STORAGE getSimOperator	-	Com.android.vending.BILLING INTERNET READ_PHONE_STATE	SYSTEM	Clean	Whitelisted
2.	YARA.STRINGMATCH .mediaRecorderStart GetLastKnownLocation getLatitude/Longitude	INTERNET READ/WRITE_EXTERNAL_STORAGE RECORD_AUDIO CAMERA getInstallPackage getExternalStorage getExternalDir	-	INTERNET READ/WRITE_EXTERNAL_STORAGE RECORD_AUDIO CAMERA	MAIN	Clean	Suspicious
3.	-	INTERNET	-	INTERNET	MAIN	-	No threat
4.	GetLastKnownLocation getLatitude/Longitude codeInjectionPossible	INTERNET getNetworkOpenConnection	Ads	ACCESS_NETWORK_STATE	MAIN	-	Whitelisted
5.	getDeviceId getSimSerialNumber CodeInjection getLatitude/Longitude	INTERNET WAKE_LOCK GetNetworkOpenConnection getExternalStorage getExternalDir	Ads	WRITE_EXTERNAL_STORAGE	MAIN	-	Suspicious
6.	GetLastKnownLocation getLatitude/Longitude	INTERNET READ/WRITE_EXTERNAL_STORAGE WRITE_SETTINGS CHANGE_WIFI_STATE CAMERA openConnection getInstallPackage getExternalStorage getExternalDir	Ads	READ/WRITE_EXTERNAL_STORAGE CAMERA ACCESS_FINE_STATE	BOOT	Clean	Whitelisted
7.	-	-	-	-	MAIN	-	No threat
8.	getDeviceId getLatitude/Longitude YARA_STRING_MATCH	INTERNET ACCESS_FINE_LOCATION OpenConnection	-	INTERNET ACCESS_FINE_STATE	-	Clean	Suspicious
9.	getDeviceId getSimSerialNumver CodeInjection getLatitude/Longitude Malicious Sample	getRunningApplications getRunningTasks getExternalDir openConnection	Ads	SERVICE SYSTEM_ALERT BOOT CALL RECEIVE_SMS/MS READ_CONTACT	MAIN	Clean	Suspicious
10.	getDeviceId YaraStringMatch getLatitude/Longitude	getRunningTasks INTERNET WRITE_EXTERNAL_STORAGE READ_PHONE_STATE WAKE_LOCK INTERNET READ/WRITE_EXTERNAL_STORAGE openConnection getInstallPackage getExternalStorage getExternalDir	Ads	WRITE_EXTERNAL_STORAGE READ_PHONE_STATE	MAIN	Clean	Whitelisted
11.	MaliciousSample getLocation codeInjection	getRunningTasks INTERNET WRITE_EXTERNAL_STORAGE READ_PHONE_STATE	Ads	ACCESS_FINE_LOCATION ACCESS_COARSE_LOCATION WRITE/READ_EXT	PKG-ADDED CONNECTIVITY_CHANGE	Clean	Malicious

		WAKE_LOCK INTERNET READ/WRITE_EXTERNAL_STORAGE openConnection getInstallPackage getExternalStorage getExternalDir		EXTERNAL_STORAGE READ_PHONE_STATE	TIME_SE T		
12.	getDeviceId getLatitude/Longitude	getRunningTasks INTERNET WRITE_EXTERNAL_STORAGE READ_PHONE_STATE WAKE_LOCK INTERNET READ/WRITE_EXTERNAL_STORAGE openConnection getInstallPackage getExternalStorage getExternalDir	Ads	INTERNET WAKE_LOCK	-	-	Whitelisted
13.	CodeInjection getLatitude/Longitude	INTERNET OpenConnection getNetworkOperator	Ads	-	-	-	No threat
14.	-	getRunningAppprocess getRunningtasks openConnection INTERNET ACCESS_COARSE_LOCATION ACCESS_FINE_LOCATION	Ads	ACCESS_FINE_LOCATION ACCESS_COARSE_LOCATION	MAIN	Clean	Whitelisted
15.	CodeInjection getLatitude/Longitude	CAMERA WRITE_EXTERNAL_STORAGE INTERNET GetNetworkOperator openConnection	Ads	CAMERA WRITE_EXTERNAL_STORAGE	-	-	No threat
16.	startRecording CodeInjection getLatitude/Longitude MaliciousSample getLastKnownLocation	getRunningTasks INTERNET WRITE_EXTERNAL_STORAGE RECORD_AUDIO openConnection getInstallPackage getExternalDir getNetworkOperator	Ads	ACCESS_FINE_LOCATION WRITE/READ_EXTERNAL_STORAGE RECORD_AUDIO ACCESS_WIFI_STATE INTERNET	SYSTEM	Clean	Malicious
17.	-	INTERNET WRITE_EXTERNAL_STORAGE WRITE_CALENDER WAKE_LOCK USE_CREDENTIALS SYSTEM_ALERT_WINDOW GetExternalDir openConnection	-	WRITE_EXTERNAL_STORAGE	MAIN RECEIVE BOOT	Clean	Whitelisted
18.	getDeviceId BOOT_COMPLETED getLatitude/Longitude getLastKnownLocation	getMACAddress INTERNET boot WRITE_EXTERNAL_STORAGE READ_PHONE_STATE WAKE_LOCK openConnection getInstallPackage getExternalStorage getExternalDir	-	WRITE_EXTERNAL_STORAGE	POWER BOOT MAIN CONNECTIVITY	Clean	Suspicious
19.	getDeviceId Malicious Sample BOOT_COMPLETED	getRunningTasks INTERNET WRITE_EXTERNAL_STORAGE READ_PHONE_STATE WAKE_LOCK openConnection	-	WRITE_EXTERNAL_STORAGE PROCESS_OUTGOING_CALL	MAIN BOOT PKG_ADDED PKG_REMOVED	Clean	Malicious

		getInstallPackage getExternalStorage getExternalDir			CONNEC TIVITY		
20.	getDeviceId	-	-	ACCESS_COARSE_ LOCATION CAMERA INTERNET WRITE/READ_EXT ERNAL_STORAGE READ_PHONE_ST ATE	MAIN ROOT	-	Malicious
21.	-	GetExternaStorageDir WRITE_EXTERNAL_STO RAGE	-	-	-	-	Whitelisted
22.	getDeviceId getLastKnownLocation getCellLocation	getInstalledPackages READ_PHONE_STATE INTERNET WAKE_LOCK openConnection setSimOperatorNumber	-	READ_PHONE_ST ATE	MAIN RECEIVE	Clean	Suspicious
23.	getDeviceId getLineNumber	getRunningtasks INTERNET ACCESS_FINE_LOCATI ON WRITE_EXTERNAL_STO RAGE CAMERA ACCESS_COARSE_LOC ATIOM READ_PHONE_STATE WAKE_LOCK GET_TASKS openConnection getMACAddress	-	ACCESS_FINE_LO CATION ACCESS_COARSE_ LOCATION WRITE/READ_EXT ERNAL_STORAGE READ_PHONE_ST ATE CAMERA	MAIN	Clean	No threat
24.	getLatitude/Longitude codeInjectionPossible	INTERNET getNetworkOp openConnection	Ads	INTERNET	-	-	Whitelisted
25.	CodeInjection getLatitude/Longitude	getRunningTasks INTERNET WRITE_EXTERNALDIRE CTORY getNetworkOperator openConnection	Ads	-	MAIN RECEIVE	Clean	Whitelisted
26.	getDeviceId BOOT_COMPLETED GetLastKnownLocation CodeInjection	getRunningAppProcess WAKW_LOCK INTERNET BOOT_COMPLETED READ_PHONE_STATE WRITE_EXTERNAL_STO RAGE CHANGE_WIFI_STATE openConnection	Ads	READ?WRITE_PH ONE_STATE ACCESS_FINE_LO CATION	MAIN CONNEC TIVITY POWER BOOT	-	Whitelisted
27.	-	openConnection getRunningTasks setRuningAppProcess INTERNET WRITE_EXTERNAL_STO RAGE READ_PHONE_STATE	Ads	WRITE/READ_EXT ERNAL_STORAGE	MAIN PKG_AD DED	--	No threat
28.	getDeviceId getLatitude/Longitude	getRUNningTasks	Ads	ACCESS_FINE_LO CATION ACCESS_COARSE_ LOCATION INTERNET WRITE/READ_EXT ERNAL_STORAGE READ_PHONE_ST ATE	MAIN PKG_AD DED	-	No threat
29.	sendTextMessage	INTERNET WAKE_LOCK	-	-	-	-	No threat
30.	CALL_PHONE SMS_RECIEVED CodeInjection	getRunningTasks INTERNET WRITE_EXTERNAL_STO RAGE	-	SEND_SMS ACCESS_COARSE_ LOCATION CAMERA	ROOT	Clean	Whitelisted

		READ_PHONE_STATE WAKE_LOCK INTERNET READ/WRITE_EXTERNAL_STORAGE openConnection getInstallPackage getExternalStorage getExternalDir		INTERNET READ_CONTACT MANAGE_ACCOUNTS READ_SMS			
31.	getLongitude/Latitude codeInjection	getRunningTasks INTERNET GetRunningAppProcess WRITE_EXTERNAL_STORAGE getExternalStorageDir openConnection	Ads	READ/WRITE_EXTERNAL_STORAGE	MAIN	Clean	Suspicious
32.	getDeviceId getLatitude/Longitude	INTERNET MODIFY_AUDIO_SETTINGS READ_PHONE_STATE WAKE_LOCK WRITE_EXTERNAL_STORAGE BLUETOOTH USE_CREDENTIALS MANAGE_ACCOUNTS	-	READ_PHONE_STATE READ/WRITE_EXTERNAL_STORAGE	MAIN	Clean	Whitelisted
33.	CodeInjection YARA STRING MATCH	getRunningAppprocess getRunningtasks openConnection INTERNET WRITE_EXTERNAL_STORAGE READ_PHONE_STATE GetExternalStorageDir	Ads	CAMERA WRITE_EXTERNAL_STORAGE READ_PHONE_STATE	MAIN EDIT VIEW CREATE_FILE	Clean	Whitelisted
34.	-	getRunningAppprocess getExternalStorageDir openConnection connect	-	-	-	Clean	Whitelisted
35.	YARA STRING MATCH getLatitude/Longitude mediaRecorder.start	getInstalledPackages INTERNET WRITE_EXTERNAL_STORAGE GetExternalStorageDir openConnection	Ads	WRITE_EXTERNAL_STORAGE	MAIN PKG_ADDED	Clean	Whitelisted
36.	getDeviceId getLastKnownLocation	getInstalledPackages INTERNET WRITE_EXTERNAL_STORAGE getExternalStorageDir openConnection	-	READ_PHONE_STATE WRITE_EXTERNAL_STORAGE ACCESS_FINE_LOCATION	MAIN PKG_ADDED	Clean	Suspicious
37.	INTERNET	INTERNET	Ads	INTERNET	MAIN	-	Whitelisted
38.	-	-	-	-	-	-	No threat
39.	getDeviceId Malicious Sample getLongitude/Latitude BOOT_COMPLETED CHANGE_WIFI_STATE READ_PHONE_STATE RECORD_AUDIO WAKE_LOCK SYSTEM_ALERT_WINDOW openConnection	getRunningAppProcesses INTERNET WRITE_EXTERNAL_STORAGE ACCESS_FINE_LOCATION	-	ACCESS_FINE_LOCATION ACCESS_COARSE_LOCATION WRITE/READ_EXTERNAL_STORAGE RECORD_AUDIO READ_PHONE_STATE	BATTERY_CHARGED USER_PRESENT	Clean	Malicious
40.	getDeviceId getLineNumber INTERNET	openConnection getInstalledApps INTERNET WRITE_EXTERNAL_STORAGE READ_PHONE_STATE	Ads	INTERNET WRITE_EXTERNAL_STORAGE READ_PHONE_STATE	MAIN VIEW	-	Whitelisted