A Review Paper on Android Malwares Analysis, Detection And Security

Ketaki A. Pattani[1], Prof. B. V. Buddhadev[2]
[1]Dept of CSE
[2]Professor, Dept of CSE
[1, 2] GEC, Gandhinagar, Gujarat, India

Abstract- *Today, Android OS powers about hundreds of millions of people who are over smart phones covering 190 countries as suggested by Official Android Market. This makes Android the best medium to bridge people but the same bridge may even lead to insecurity and penetration. The current scenario shows that Android covers more than 80 percent of the mobile market.[2] Also, since Android supports application deployment, this brings forth great chances of being attacked by malwares. The current status of malware development notices a great rise of about 7.11 million in near future. Thereby, this created a requirement of detecting and developing antimalware techniques. Thus, the current paper is a threefold survey dealing with Android malware categorizations, their methods of detection and antimalware techniques for security of android devices.*

Keywords– Android Security, Malwares in Android, Malware Detection, Antimalware, Signature.

I. INTRODUCTION

The current research over Android use and market share suggests that Android market covers a wide range of smart phones.[1] As per International Data Corporation IDC share 2018 First Quarter analysis Android shares about 84.8% of the total market. Whereas, iOS covers 15.1% and others cover 0.1% of the total share.[2] Android hosts millions of applications everyday all of which may not be equally secure. This creates a threat of undetected malwares being transferred throughout the network via such applications. The malware development analysis suggests that the rate of malware development is increasing at a high pace. There are more than 600 million malwares developed and forecasted in 2016 and among them over mobile platform Android holds a leading position with high risk malwares. This creates a requirement for the Android market to develop combatant constructs that resist such malwares and their effects. Such combatants are called anti-malwares in technical terminology. Thereby, to get rid of such Android malwares, one needs to have critical knowledge about Malware and their uses. As an example of malware an application say 'Messaging Application' takes permissions related to the messaging interfaces. Now it may be malware infected and may send sensitive information towards a sink. Such an attack is said to be permissions based attack and may evade the personal information of the user.

In this paper analysis of various such types of attacks as mentioned in the above example is done. It represents various types of malwares in section II, Android malware penetration techniques in section III and Android malware detection techniques in section IV. Finally section V shows the analysis of these techniques and later sections provide a solution and its future scope of improvement.

II. ANDROID MALWARE ANALYSIS

Here, there is keen analysis of malware categories, malware characteristics and standardized approaches adopted to evade security. The different types of malwares are categorized based on their operations and working. Thereby, a group of malwares performing certain kind of operation are considered to be in the same family also known as Malware Family. Following is the specification of various Malware Families with examples cited with each of them showing their effect on Android devices.

1. *Trojan:* The keyword Trojan is termed from the ancient story of 'Trojan Horse'. The Trojan does the activity of stealing user's confidential data without user even being notified about it. The stolen data is then leaked to the attacking sources.[1] Example: FakeNetflix, Zsone, FakePlayer etc.
2. *Backdoors:* There are certain malwares that require root privileges as they access highly secure system resources. Backdoors are meant to provide root privileges to the malware and take control over the device so that it cannot be interrupted. Example: Zimperlich, Exploid and RageAgainstTheCage.
3. *Worms:* Creates multiple copies of itself and distributes them over the network. Another common use of worm that everyone is making is from a malware application a worm is created which infects all the other applications by its copies. Example: Android.Obad.OS is a Bluetooth worm.
4. *Spyware*: Spyware containing App usually appears to be gentle but actually monitors user's confidential data, events, logs etc. They may also install malicious payload

and transfer secret information to attacker's sink. Example: NickSpy, GPSSpy.

5. *Botnet:* It is a network of Android devices each of which has a number of bots. They are victims of attackers and are usually used for Distributed DOS attack to be done on some external server. Example: Gemini.
6. *Ransom ware*: They do not allow user to access their own data until some ransom is paid to them. Example: Fake Defender acts as Avast Antivirus and locks the phone until the ransom is paid.
7. *Risk wares*: They are the possible risks created over the data or system but not definite risks on installation. Some regular applications may also behave as risk wares. It may edit, delete or modify the data or permissions.

Type	Detected Installation 2016	Detected Installation 2017
Trojan	21.51%	30.28%
Backdoors	0.94%	0.87%
Worms	37.02%	11.56%
Viruses	30.04%	49.20%
Botnet	0.39%	0.37%
Ransomware	0.86%	0.87%
Riskware	3.02%	2.24%

The statistics are the overall malware distribution detected in 2015 and 2016[3]. There may be certain malware not detected then. Therefore, here the analysis for Android suggests that Android devices are on verge of high risk and alarming that high end security system must be developed for protection. But before reaching towards solutions its penetration techniques must be taken into consideration which is explained as below.

III. MALWARE PENERATION TECHNIQUES

The penetration techniques are used by the attackers to bypass the various analysis.[4] Following are the different types of penetration techniques:

1. *Repackaging with malware*: This technique develops the view of some popular app by disassembling the application and then adds its malware content to the app and again reassembles it and puts it on less monitored 3rd party market. For example: Amazon Application Store.
2. *Drive By Download*: Here, as user tries to download any resource unintentionally a malware is downloaded in the background without the user being notified. Developers use the 'Non-Compatible Android Trojan' to perform this task.
3. *Dynamic Download of Payloads*: Here, the encrypted payloads are executed at runtime to perform malicious activities. Certain malwares are also used to download these payloads at run time in order to fool static analysis tool.
4. *Malware with Stealing Techniques*: Since direct analysis of android app causes battery and resources issues, certain techniques are used to obfuscate anti-malwares as key permutation obfuscation approach, dynamic loading of data for obfuscation, native code execution and stealth of data etc. to attack victim's device.

IV. ANDROID MALWARE DETECTION

Majorly, every scenario suggests two approaches categorized at a higher level to detect the Android malwares as: the Static and the Dynamic Approach. Even the defensive anti-malware gets classified based on the mentioned two standard categories.[6]

I. Static Analysis

Here, the maliciousness of malware is checked by analyzing source code without executing it. Example: Certain behavior of the system seen on occurrence of particular event. They are further categorized as explained

1. Signature Based Approach

The approach analysis sementic patterns and creates a unique signature for each malware. So, if new application comes having similar type of malware, it is detected using signature.[5] Such methods are very easy to obfuscate as they do not identify unseen malwares.

EXAMPLES:

a. AndroSimilar by Faruki

Usually malwares are added upon existing applications and repackaging is done. Mentioned tool finds such obfuscating malware contained tools. The tool gives more than 60% true outcomes with correct detections.

b. Droid Analytics

The tool uses the same signature base approach but extracts and analysis every application at their operation-code or instruction syllable level. It does this by generating three level signatures. The levels are designated as Method Level using code tracing at Application Programming Interface, class level and application level. The similarity based score provided on detection does not prove to be fully accurate and may have false positives in results.

Limitations of Signature Based Approach

The most vulnerable aspect of this method is that it cannot detect unknown or unseen malwares. Further, due to pre-defined database of signatures there may be results with undetected outcome.

2. Permission Based Approach

The base structure of Android consists of AndroidManifest.xml which consists of all the permissions that are needed for an application. Permissions granted to an application are actually more than required most of the times.

EXAMPLES:

a. Stopaway

There may be applications requesting hazardous combinations of permissions more than actually required set of permissions. The work here does analyze the code statically to track API calls and permissions to determine the level of vulnerability. Analysis shows one-third results to be having over-privileged permissions in a total of 940. However, the API calls made or executed by applications with java reflections remain undetected.

b. Extract Manifest

Given approach shows the malignancy or vulnerability score based on the analysis of manifest file. It is a light weight proposal working over extracted information and its comparison with the list of keywords within the proposed method. Based on the score the application is termed as malicious or trustable.

c. Analysis of permissions

As per the known fact that the applications ask for permissions which prove o be more than what is actually required. So, analysis of permissions develops comparison of the requested as well as the required permission set. This is done by analysis of features and characteristic behavior of the applications. Once analysis is done, labeling is followed into three different classes or types. The three involve site based labeling as the first; scanner based labeling as second and third as mixed labeling. Thereafter, samples are listed to three different datasets and algorithms such as the *Naive Bayes*, *AdaBoost, etc.* are utilized to evaluate its behavior. Finally, the malwares are determined to be present or absent.

d. PUMA

One another method of malware detection is PUMA which again analyzes the permission requirement of application, but uses tag based analysis for the same. Tags determining permissions in AndroidManifest.xml file are helpful for the process. Classifier algorithms play a vital role and are applied on 357 trusted applications and 249 harmful applications. The consequence is a high end detection rate but also has heavy false positives. So, it cannot be used efficiently and requires other dynamic ways of detection to be implemented.

e. Security Distance Model

Also abbreviated as the SD Model, given application focuses that one permission alone cannot pose threat to the security of the system. Instead there may be more than one combination or group of permissions that may be hazardous and threatening to the system. Thus, the SD Model classifies the application based on set of permissions. And also terms some of them as malicious. The work proceeds by classification of permission requested into four different groups and assigning them threat points as TP-0 as Safe, TP-1 as Normal, TP-5 as Dangerous and TP-25 as Serious level. This certainly determines the possible attack level and threat.

f. KIRIN

KIRIN is a tool developed by Enck [36], useful for certification. This is a very light weight process done at the installation phase. Certain rules for the security of data are pre-defined and also comparison with application asked permissions is done. Undertaken application is considered as malicious on account of failure in regulations defined. But, the main issue with KIRIN is when it determines certain trusted applications also as vulnerable which may not be reliable for the usage.

g. DroidMat

It is an application using K-means clustering algorithm for better malware detection as well as determining applications o be benign or possibly vulnerable. System here extracts manifest based information involving permissions, communication in intents and tracing API calls. Tool uses KNN algorithm for analysis. However, DroidMat cannot determine dynamically loaded malicious activities as it functions statically.

Limitations in Analysis of Permissions

The very outer approach for analysis of malignity is permission analysis by portraying the list of vulnerable permission combinations and then checking for their presence. This may give a huge range of false positives as there is a very thin line between the permissions requested and actually required for benign and malicious applications. This represents that here must be another level over it for analysis of malignity.

3. Dalvik Bytecode Analysis:

Functionally applications developed in Android have back-end in java and are further converted to Dalvik code which is a VM based on registry. Such an analysis of bytecode determines the functionality and feature based behavior of the application. Also, the control flow analysis and the data flow analysis shows highly vulnerable functionalities of apps.

EXAMPLES:

a. SCANDAL

Developed by Jimyung Kim, the tool creates analysis of Dalvik Bytecode and then detects privacy leakage. It detected 11 out of 90 Android applications malicious. It analyzes all possible paths from source to remote server by branch based approach, method invocation for detection and approach involving jump instructions tracking. It does not support reflections calls so they must be manually written.

b. Formalization with reflection

Karlsen has analyzed over 1700 applications and Dalvik Bytecode is formalized using java reflection. This does control the malignity by detection of control flow and data flow based vulnerabilities. System requires betterment in reflection and concurrency handling in application. But, it has enhanced dynamic dispatch.

c. DroidMOSS

It generated finger print for each tool by getting Dalvik Bytecode Sequence and Developer Information to detect repackaged information. Hence it has unique signatures and checks. But the issue is it must have original application in its database.

d. DroidAPI Miner

Use of KNN algorithm for API call tracking, analysis of hazardous parameters and information analysis using bytecode is done. Tool uses Androguard and gives upto 99% accurate results and only 2.2% false positives.

e. SCandroid

Fuchs the developer, has developed the tool that does static analysis at the installation time and maps data flow based analysis. Thereby, considering data flow analysis at run-time and behavioral aspects along with permissions it dignity is determined.

Limitations of Dalvik Byte Code

The major disadvantage is that the analysis occurs at the instruction level. This is very time consuming and also high storage is required which is unaffordable by Android.

II. Dynamic Approach

This approach examines the application during execution so that they can even detect malwares with obfuscating approaches.

Suggested by Egele, the dynamic analysis methods require certain mode resources but are better comparatively.

1. Anomaly based detection

It refers to the behavior based analysis of applications and is also known as Behavioral Malware Detection. Anomaly based detection may function two phases : training phase and detection phase. In the training phase the detector tries to learn from its obvious behaviors. The major advantage of anomaly based detection is that it can detect the zero day attacks.

EXAMPLES:

a. CrowDroid

It is a tool for Dynamic Analysis. The details of the application are collected by trace based tool. Crowdsourcing app generates a logging content file and transports it to

another remote server where certain algorithm for cluster formation is used. Generated are the results accumulated on database. It may also classify safe benign application as a malware if there are heavy system calls.

b. Shabtai

Shabtai is a tool performing detection based on the behavioral analysis of application run.hus changes in certain measures are continuously checked for and also machine learning is used to be applied so as to improve its status and detecting capability. Thereby, detection of benign or vulnerable application is done.

c. AntiMalDroid

Zhao the publisher determines dynamic analysis of application to track their execution using SVM algorithm. Firstly, the analysis is done for the determination of benign or the malicious application and then they are put into learning module after which signature is prepared. The signature is the basis and is used every time to check any application for vulnerability.

Limitations of Anomaly Based Detection

There are false positives when even a safe application shows uncertain behaviors as battery drainage.

2. Taint Analysis

Tainting here is dynamic and allows system wide information flow tracking system. This may include tracking and tainting of all system wide resources.

EXAMPLES:

a. TaintDroid

It provides system-wide information flow tracking for Android. It can simultaneously track multiple sources of sensitive data such as camera, GPS and microphone etc. and identify the data leakage in third party developer apps. It labels the sensitive data and keeps track of that data and app when tainted data leaves moves from the device to sink.

b. XmanDroid

There are certain issues with the TaintDroid as it cannot adequately address certain attacks. XManDroid analyzes communication links among applications and ensures they comply to a desired system policy. XManDroid can prevent privilege escalation attacks including collusion attacks (e.g., Soundcomber) that exploit the covert channels provided by the Android's core application. Further, there is integration of a new concept for storing the decisions made by XManDroid which can be directly integrated in the standard permission framework of Android.

3. ***Emulation based detection***

In emulation based technique, system can detect the behavior of malware as well as the sequence of malware. This technique is used to minimize the time of detection. It is also used to detect both the polymorphic malware and even the metamorphic malware.

EXAMPLES:

a. *DroidScope*

Developed by Yan, it is based on Virtual Machine Introspection. *DroidScope* monitors the whole operating system by staying out of the execution environment and thus have more privileges than the malware programs. It also monitors the Dalvik semantics thus the privilege escalation attacks on kernel can also be detected. It is built upon QEMU. *DroidDream* and *DroidKungFu* were detected with this technique.

b. *Android Application Sandbox*

Developed by Blaising, it performs both static and dynamic analysis. It first extracts the .dex file into human readable form and then performs static analysis on application. Then it analyzes the low level interactions with system by execution of application in isolated sandbox environment. Actions of application are limited to sandbox due to security policy and do not affect the data on device. It uses Money tool to dynamically analyze the application behavior which randomly generates the user events like touches, clicks and gestures etc. it cannot detect the new malware types.

V. CONCLUSION

The analysis of the current scenario according to IDC[2] suggests that current mobile market has high influence of Android. Also, the Android Malware is increasing exponentially its power. But yet there are many covert channels and evasive techniques not successfully defended by the anti-malwares. These aspects discover great scope of development as relatively less progress is seen in this area. Anti-malwares should combine static as well as dynamic approach and defend against such newly developing attacks. Hence, the paper not only shows the status of existing

malwares but also concludes that anti-malwares need to cope up with the everyday growing vulnerabilities with appropriate techniques.

VI. FUTURE WORK

The two upthrusting fields in Information Security are Evasion and Covert Channels that are not defended much from security point of view. There are many covert channels as Settings of volume, vibration etc. used to leak privacy based data e.g. contacts to sink. One such Covert Channel is Ultrasound.[7] Any mobile device having speakers is capable of producing frequencies considerably high for major humans to hear. Generated ultrasound can be received by a microphone on the same device or on another device. Based on this the ultrasound can also be flooded with private data and it can be easily migrated to the sink. Evasion here helps to hide the data flow of the data sent along with covert channels so that it goes unnoticed to the users as well as propriety anti-malwares. Whereas, covert channels establish a secure and unnoticed transmission channel. Such an attack can unnoticeably evade the security of any user. Relative defense techniques must also be published.

REFERENCES

[1] Y. Zhou and X. Jiang, “Dissecting Android Malware: Characterization and Evolution,” 2012 IEEE Symp. Secur. Priv., no. 4, pp. 95–109, 2012.

[2] “IDC: Smartphone OS Market Share 2022, 2021, 2020, 2019, and 2018.” [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp. [Accessed: 20-Sep-2018].

[3] “Malwares types and their market share in the years 2015 and 2016.”. Available at: https://securelist.com/mobile-malware-evolution-2016/77681/ . [Accessed: 20-Dec-2017].

[4] Parvez Faruki Ammar Bharmal Vijay Laxmi Vijay Ganmoor Manoj Singh Gaur Mauro Conti Muttukrishnan Rajarajan "Android security: a survey of issues malware penetration and defenses" *IEEE communications surveys & tutorialsno*. 2 at http://ieeexplore.ieee.org/document/6999911/

[5] P. Faruki, V. Ganmoor, V. Laxmi, M. S. Gaur, and A. Bharmal, “AndroSimilar: Robust Statistical Feature Signature for Android Malware Detection,” Proc. 6th Int. Conf. Secur. Inf. Networks, pp. 152–159, 2013.

[6] R. Raveendranath, V. Rajamani, A. J. Babu, and S. K. Datta, “Android malware attacks and countermeasures: Current and future directions,” 2014 Int. Conf. Control. Instrumentation, Commun. Comput. Technol., pp. 137–143, 2014.

[7] D. Arp E. Quiring C. Wressnegger K. Rieck "Privacy threats through ultrasonic side channels on mobile devices" Proc. IEEE Eur. Symp. Secur. Privacy pp. 35-47 Jul. 2017.