A perspective study on Malware detection and protection, A review

Diptiben H. Gillani

***Abstract*—**Android has become the most popular smartphone operating system. This rapidly increasing adoption of Android has resulted in significant increase in the number of malwares when compared with previous years. There exist lots of antimalware programs which are designed to effectively protect the users' sensitive data in mobile systems from such attacks. In this paper, our contribution is twofold. Firstly, we have analyzed the Android malwares and their penetration techniques used for attacking the systems and antivirus programs that act against malwares to protect Android systems. We categorize many of the most recent antimalware techniques on the basis of their detection methods. We aim to provide an easy and concise view of the malware detection and protection mechanisms and deduce their benefits and limitations. Secondly, we have forecast Android market trends for the year up to 2018 and provide a unique hybrid security solution and take into account both the static and dynamic analysis an android application.

Keywords: Android; Permissions; Signature

INTRODUCTION

Since 2008, the rate of smartphone adoption has increased tremendously. Smartphones provide different connectivity options such as Wi-Fi, GSM, GPS, CDMA and Bluetooth etc. which make them a ubiquitous device. Google says, 1.3 million Android devices are being activated each day [1]. Android operating system left its competitors far behind by capturing more than 78% of total market share in 2013 [2]. Gartner report 2013 of smartphone sales shows that there is 42.3% increase in sales of smartphones in comparison with 2012. According to International data corporation IDC, Android OS dominates with 82.8% of total market shares in 2Q 2015 [3]. Figure 1 shows the market shares of Android operating system on yearly basis. It could be observed that Android has become the most widely used operating system over the years. Android platform offers sophisticated functionalities at very low cost and has become the most popular operating system for handheld devices. Apart from the Android popularity, it has become the main target for attackers and malware developers. The official Android market hosts millions of applications that are being downloaded by the users in a large number everyday [4]. Android offers an open market model where no any application is verified by any security expert and this makes Android an easy target for developers to

A. Static Approach

Static approach is a way to check functionalities and maliciousness of an application by disassembling and analyzing its source code, without executing the application. It is useful for finding malicious behaviors that may not operate until the particular condition occurs.

1) Signature Based Approach

Signature based malware detection methods are commonly used by commercial antimalware products. This method extracts the semantic patterns and creates a unique signature. A program is classified as a malware if its signature matches with existing malware families" signatures. The major drawback of signature based detection is that it can be easily circumvented by code obfuscation because it can only identify the existing malwares and fails against the unseen variants of malwares. It needs immediate update of malware variants as they are detected. Faruki *et al*. [26] proposed *AndroSimilar*, a robust statistical signature method to detect the unknown variants of existing malwares that are usually generated by using repackaging and code obfuscation techniques. It generates the variable length signature for the application under test and compares it with the signatures in AndroSimilar malware database and identify the app as malware and benign on the basis of similarity percentage. Authors tested the AndroSimilar against 1260 apps among which 6779 apps were Google Play apps and 545 apps were from third party app store. They also used code obfuscation techniques such as method renaming, string encryption, control flow obfuscation and junk method insertion techniques to change the signature of the code and tested the effectiveness of AndroSimilar against 426 samples. The solution detected more than 60% samples correctly. AndroSimilar compares the signatures of the applications in order to distinct between the malwares and benign apps but it has limited signature database as compared to the other antivirus solutions. So any unseen malwares will remain undetected. Also the similarity percentage creates the false positives as it may classify the clean apps as malicious on the basis of percentage. *DroidAnalytics* [27] is a signature based analytic system which extract and analyze the apps at op-code level. It not only generates the signature but also associate the malware with existing malwares after identifying the malicious content. It generates 3 level signatures. First it generates signature at method level by API call tracing then combining all the signatures of methods in a class it generates the class level signatures and at third level it generates the application signature by combining the signatures of the classes in the application. Authors have used DroidAnalytics to detect 2,494 malware samples from 102 malware families and 342 repackaged malwares from other six malware families. The limitations of this method includes, it classifies the apps as malware on the basis of classes mostly used by malware families but during experiment they found some signatures that are used by both the legitimate apps and malwares. Also the similarity score used for detection of repackaged malwares do not provide 100% solution or it may also provide false positive, classify the legitimate app as malware.

- *Limitation of Signature Based Detection:* Although signature based detection is very efficient for known malwares but it cannot detect the unknown malware types. Also because of limited signature database most of the malwares remain undetected.

2) Permission Based Analysis:

In Android system, permissions requested by the app plays a vital role in governing the access rights. By default, apps have no permission to access the user‟ data and effect the system security. During installation, user must allow the app to access all the resources requested by the app. Developers must mention the permissions requested for the resources in the AndroidManifest.xml file. But all declared permissions are not necessarily the required permissions for that specific application Ref. [28] has shown that most of the time developers have declared the permissions that are not actually required by the application which makes it difficult to detect the malicious behavior of application. Antimalware analyzes the Android Manifest.xml file where all the permissions for the resources required by the app are mentioned. *Stowaway* [28] exposes the permission over privilege problem in Android where an app requests more permissions than it actually uses. *Stowaway* performs static analysis to determine the API calls invoked by the application and then it maps the permissions required by the API calls. They found that one third applications are over privileged among 940 Android application samples. It cannot resolve the API calls invoked by applications with the use of java reflections. In [29], authors have proposed a light weight malware detection mechanism which only analyze the manifest file and extract the information such as permissions, intent filters (action, category and priority), process name and number of redefined permissions to detect the malicious behavior of an application. After extracting such information, they compare it with the keyword list provide in the proposed method and then calculate the malignancy score. They used *Weka* [30] which is a data mining tool for calculation of threshold value. At last they compare the malignancy score with threshold value and classify the app as malware if malignancy score exceeds threshold value. They have used 365 samples to test the efficiency of proposed solution and the solution provides 90% accurate detection. It is cost saving mechanism as it only includes the analysis of manifest file and can be implemented in other detection architectures easily to detect malwares efficiently. Also it can detect even those malwares that remain undetected by signature based detection method. This proposed solution is limited to manifest file information. Also it cannot detect the adware samples. C. Y. Haung *et al*. [31] proposed a method for better detection of permission based malware detection which includes the analysis of both requested and required permissions as most of the time malware authors declare more permissions in the manifest file than they actually require for the application. Also it analyses the easy to retrieve features and then labels the application as benign or malware. Three different labeling types are used for this purpose which includes site based labeling; scanner based labeling and mixed labeling. In site based labeling it labels the app as benign if it is downloaded from Google official app market and if it is downloaded from some malicious source then the app is labeled as malicious. In the second labeling scheme, if the antivirus scanner declares the app

as benign the app is label as benign and same for the malware case. In the mixed labeling the app is labeled on the basis of both site based and scanner based labels. After labeling all the samples are divided into three datasets and requested permissions of these datasets are analyzed by the machine learning algorithms such as *Naive Bayes, AdaBoost, Support Vector Machine and Decision Tree* [32]. On the basis of results generated by these classifiers we can evaluate the performance of permission based detection method. in [31] authors have performed experiment on data set of 124,769 benign and 480 malicious apps

They analyzed the performance of permission based detection of malware and showed that more than 81% of malicious apps samples can be detected by the permission based detection method. Proposed method provides the quick filter for malware detection but the performance values generated by the classifiers are not perfect and we cannot completely rely on those results. Sanz Borja *et al*. [33] presented *PUMA* for detection of malicious apps by analyzing the requested permissions for application. They used permission tags such as <uses-permission> and <uses-features> present in AndroidManifest.xml file to analyze the malicious behavior of apps and applied different classifier algorithms on dataset of 357 benign apps and 249 malicious apps. The solution provides high detection rate but results generated have high false positives rate also it is not adequate for efficient detection of malware it still requires information related to other features and dynamic analysis. Shin *et al*. [34] used a state machine based approach and formally analyze the permission based Android security model. They also verified that the specified system satisfy the security property. Tang, Wei *et al*. [35] proposed a Security Distance Model for mitigation of Android malware. Security Distance Model is based on the concept that not a single permission is enough for an application to threaten the security of Android devices. For example an application requesting permission READ_PHONE_STATE can access the phone number and IMEI but it cannot move data out of the device. There must be a combination of permissions to affect the security model of device such as INTERNET permission allows to concept the device with the network and will be needed to move data to some remote server. The SD measure the dangerous level of application on the basis of permissions requested by the app. Authors classify the combinations of permissions into four groups and assigned threat points (TP) to each group such as TP-0, 1, 5 and 25 to Safe SD, Normal SD, Dangerous SD and Severe SD. Before the installation of new application it calculates the threat point from the combination of permissions requested by the application. That helps the user to get aware of more dangerous permissions while installation of app. It can easily detect the unknown malwares with very high threat points. They found 500 threat points for the Geinimi malware which is a very clear variation from benign apps. A limitation of this solution includes that applications with threat points between 50 and 100 are not easy to identify as benign and malware. They could be the benign apps with such permission combinations or malwares. Enck *et al*. [36] developed *KIRIN*, a tool that provides light weight certification at installation time. It defines the security rules and simply compares the requested permissions of app with its security rules and certifies the app as malware if it fails to pass all the security rules. The installation of app is aborted if the app is attributed as malware. Authors have tested 311 applications downloaded from official Android market and found that 5 applications failed to pass the specified rules. Proposed solution is light weight as it only analyzes the Menifest.xml file. The limitation of *KIRIN* includes that it may the behavior of different malware families is provided in subsequent sections.

A. Trojans

Trojans appear to a user as a Benign app [5]. In fact, they actually steal the user‟s confidential information without the user‟s knowledge. Such apps can easily get access to the browsing history, messages, contacts and device IMEI numbers etc. of victim‟s device and steal this information without the consent of user. *FakeNetflix* [10] is an example of such malwares that provide user interface identical to original Netflix app and collect the user‟s login credentials. SMS Trojans exploit the premium services to incur financial loss to the victim. *Fakeplayer* is a well-known SMS Trojan that sends messages to premium rate numbers without user awareness
[11]. *Zsone* [12] and *Android.foney* are also the examples of such SMS Trojan apps. Malwares also capture the user‟s banking information such as account number and password. *Zitmo* and *Spitmo* Trojans are designed to steal the user‟s mTANs (Mobile Transaction Authentication Number) which then complete the transactions silently [13].

B. Backdoors

Backdoors employ the root exploits to grant root privileges to the malwares and facilitate them to hide from antiviruses. *Exploid, Rageagainstthecage (RATC)* and *Zimperlich* are the top three root exploits which gain full- control of device [14]. *DroidKungFu* [15] uses root exploits, *Exploid and Rageagainstthecage,* in an encrypted form. When *DroidKungFu* executes, it first decrypts and launches the root exploits. If the root exploit succeed to gain control over device and root privilege, the malware become able to perform any operation on the device even the installation of applications keeping the user unaware of this act [16].

C. Worms

Such malwares create copies of it and distribute them over the network. For example, Bluetooth worms spread malware through the Bluetooth network by sending copies of it to the paired devices. *Android.Obad.OS* is the example of Bluetooth worm [1].

D. Spyware

Nickspy [11] and *GPSSpy* [18] are the examples of spyware apps which appear as benign app, but it actually monitors the user‟s confidential information such as messages, contacts, bank mTANs, location etc. for some undesirable consequences. Personal spywares can install the malicious payload without the victim‟s knowledge. It sends the user‟s information such as text messages, contacts etc. to the attacker who installed that software on victim‟s device [6].

E. Botnets

Botnet is a network of compromised Android devices. Botmaster, a remote server, controls the botnet through the C&C network. Geinimi [11] is one of the Android botnets.

F. Ransom wares

Ransomware prevent the user from accessing their data on device by locking the device, until ransom amount is paid. Fake Defender.B [19] is a malware that masquerades itself as avast!, an antivirus. It locks the victim‟s device and force the user to pay ransom amount to unlock the device.

Conclusion

In this paper, the malwares and their penetrations techniques is also provided and the benefits and limitations of these antimalware are deduced comprehensively. At the end, a concept of hybrid antimalware is presented which will address the limitations of existing static and dynamic approaches. In future, it is aimed to implement the proposed hybrid solution which will be a generic antimalware that will provide better security for Android devices by firstly statically analyzing the Android applications on local device and then it will perform dynamic analysis on a remote antimalware server. This will consume very small amount of memory space on the device and the battery consumption will also be low as all dynamic analysis will be performed at the remote server.

References

[1]. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "A survey of mobile malware in the wild," Proc. 1st ACM Work. Secur. Priv. smartphones Mob. devices - SPSM "11, pp. 3 – 14, 2011.

[2]. R. Fedler, J. Schütte, and M. Kulicke, "On the Effectiveness of Malware Protection on Android," p. 36, 2013.

[3]. "Mind the (Security) Gaps: The 1H 2015 Mobile Threat Landscape - Security News - Trend Micro USA." [Online]. Available: http://www.trendmicro.com/vinfo/us/security/news/mobile-safety/mind-the-security-gaps-1h-2015-mobile-threat-landscape. [Accessed: 08-Dec-2015].

[4]. 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.

[5]. Ughulu, J. Entrepreneurship as a Major Driver of Wealth Creation.

[6]. Y. Zhou and X. Jiang, "Dissecting Android Malware: Characterization and Evolution," 2012 IEEE Symp. Secur. Priv., no. 4, pp. 95–109, 2012. "Security Alert: Zsone Trojan found in Android Market | Lookout Blog." [Online]. Available: https://blog.lookout.com/blog/2011/05/11/security-alert-zsone-trojan-found-in-android-market/. [Accessed: 15-Dec-2015].

[7]. L. Davi, A. Dmitrienko, C. Liebchen, and A.-R. Sadeghi, "Over-the-Air Cross-platform Infection for Breaking mTAN-based Online Banking Authentication," Black Hat Abu Dhabi, pp. 1–12, 2012.

[8]. Ughulu, D. (2022). The role of Artificial intelligence (AI) in Starting, automating and scaling businesses for Entrepreneurs. *ScienceOpen Preprints*.

[9]. Ghelani, D., & Hua, T. K. (2022). Conceptual Framework of Web 3.0 and Impact on Marketing, Artificial Intelligence, and Blockchain. *International Journal of Information and Communication Sciences*, *7*(1), 10.

[10]. Ghelani, D., & Hua, T. K. A Perspective Review on Online Food Shop Management System and Impacts on Business.

[11]. Oak, R., Du, M., Yan, D., Takawale, H., & Amit, I. (2019, November). Malware detection on highly imbalanced data through sequence modeling. In *Proceedings of the 12th ACM Workshop on artificial intelligence and security* (pp. 37-48).